My Fantasy

שם המגיש: נעם כירם

מורה מלווה: גיא אלרם

שם בית הספר: מקיף א' גימנסיה ריאלית ראשון לציון

אפריל 2019

תוכן עניינים

מבוא

שם תלמיד: נעם כירם

שם המורה המלווה: גיא אלרם

שם האפליקציה:.My Fantasy

כללי: האפליקציה היא גרסה מצומצמת למשחק Fantasy. הפלטפורמה מאפשרת לשחקנים להרכיב לעצמם קבוצות דמיוניות משחקני N.B.A אמיתיים, ולנהל ליגות על בסיס נתונים סטטיסטיים.

קהל היעד: חובבי ליגת ה-N.B.A בשנים האחרונות.

מטרת האפליקציה:

האפליקציה היא משחק סטטיסטי, שבו שחקנים בונים קבוצת כדורסל ומתחרים זה בזה.

הסבר: האפליקציה, שתרוץ על מערכת הפעלה אנדרואיד בלבד, היא משחק שבו כל משתתף בונה לעצמו נבחרת עם חמישה שחקני N.B.A בשנה ספציפית. על המשתמש לבחור שנת N.B.A ולאחר מכן לבנות נבחרת, המורכבת משחקנים המשחקים בליגה באותה עונה. השחקן יכול לאתגר חברים נוספים, שברשותם האפליקציה, לבנות גם הם נבחרת מאותה שנה. השחקנים נבחרים לפי תפקידם על המגרש (גארד, פורוורד, סנטר) מתוך רשימה מובנית. לאחר בניית הקבוצה כל קבוצה תקבל את הניקוד שלה לפי הסטטיסטיקה של השחקנים באותה עונה, לדוגמה מספר נקודות, אסיסטים, ריבאונדים, חסימות וחטיפות, ובסופו של דבר יוכרז המנצח שקבוצתו זכתה בניקוד הכולל הגבוה ביותר.

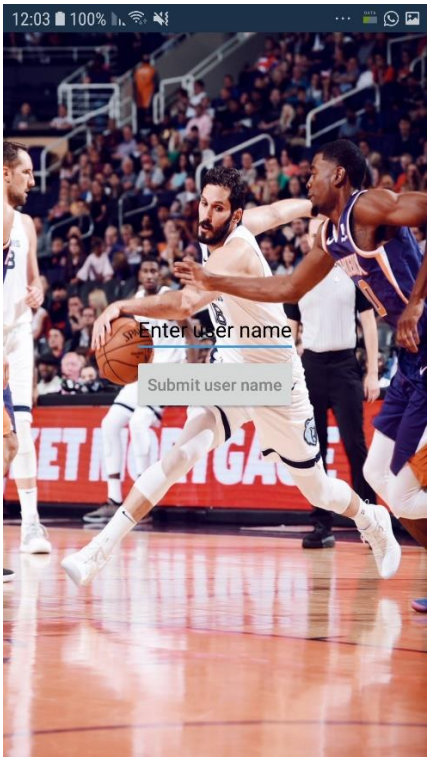
האפליקציה תיעזר בשרת אשר ייכתב בשפת פייתון. השרת ידגום את אתר ה-N.B.A בזמנים קבועים, יאחזר את הנתונים הסטטיסטיים של השחקנים וישמור אותם על השרת. הנתונים מגיעים במקור מאתר ה- N.B.A באמצעות שירותי REST. נתוני הנבחרות של משתמשי האפליקציה יישמרו גם הם על השרת, כך שהשרת יבצע את ההשוואה בין משתמשי האפליקציה.
כמו כן, השרת ישמור את מצב המשתמשים שאותגרו (באיזה שלב באפליקציה נמצא כל משתמש - אתגר נשלח / בונה קבוצה וכו').

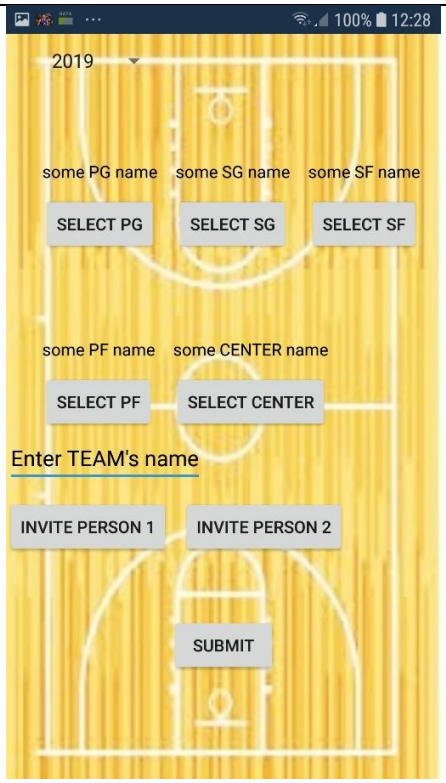
בצד הלקוח, יישמרו נתונים מינימליים המאפשרים הזדהות בפני השרת על מנת למנוע הזדהות חוזרת בכל אפליקציה. בעת בניית קבוצה, לאחר בחירת השנה, תועבר רשימת שחקנים שהיו פעילים בעונת משחקים מצד השרת ללקוח.

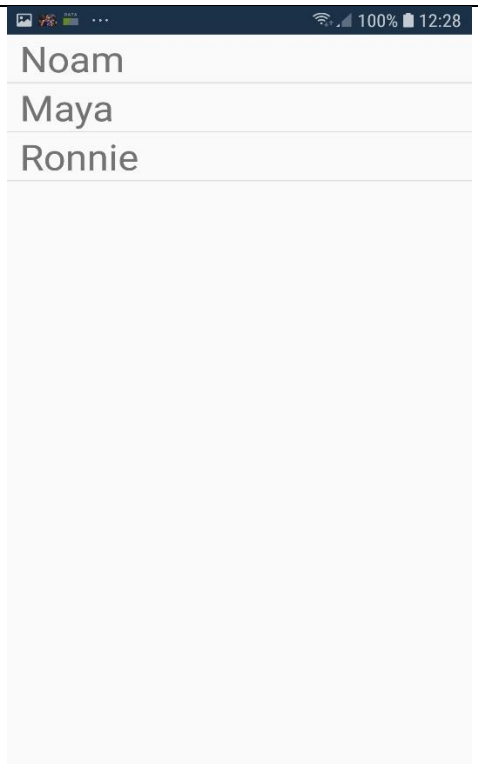
המסכים באפליקציה:

1. מסך פתיח המוצג למספר שניות.
2. מסך הרשמה למשחק (חד פעמי).
3. מסך בחירה האם להיענות לאתגרים שנשלחו ליצור אתגר.
4. מסך בניית נבחרת - בו השחקן בוחר עונת משחקים, חמישיית שחקנים מאותה עונה, והזמנת שני משתמשים נוספים במידת הצורך.
5. מסך רשימת שחקנים – מציג את השחקנים לפי עונה ועמדה מסוימת.
6. מסך בחירת משתמשים – בשביל להזמינם לאתגר חדש שנוצר.

טבלת מסכים באפליקציה

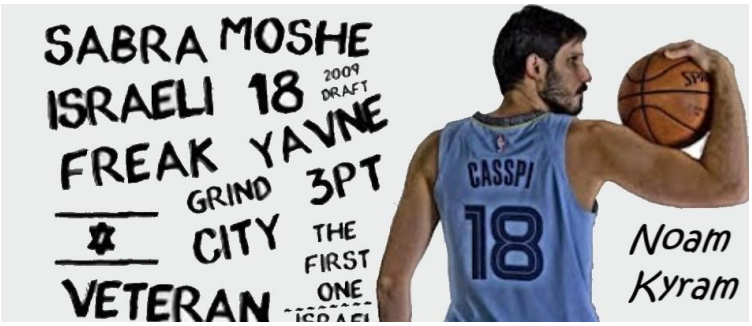
שם המסך	תכולה	תיאור	תמונה
מסך פתיח	אין תכולה במסך זה.	מסך זה מוצג למספר שניות ומוצגת התמונה.	
מסך הרשמה	תיבת טקסט שם משתמש – בה המשתמש יכניס את שם המשתמש שבו ירצה להזדהות אצל השרת כאשר נכנס לאפליקציה.	מסך הרשמה חד פעמי שהמשתמש יעשה מיד לאחר התקנת האפליקציה. המשתמש יכניס את שם המשתמש שלו, ויאשר אותו. שם המשתמש יישמר במכשיר. בכל פעם שהמשתמש נכנס לאפליקציה, ייבדק האם שם המשתמש נמצא בשרת, אחרת ייפתח מסך ההרשמה.	

מסך היענות לאתגרים	רשימה בה יוצגו כל האתגרים שנשלחו מהמשתמשים האחרים, והניקוד של הקבוצות שנבנו. כפתור של יצירת אתגר חדש.	מסך בו יצטרך המשתמש לבחור האם להיענות לאתגר שאחרים הציעו לו או ליצור אתגר משלו. בכחול מסומנים האתגרים שיצרת, בכחול מוכרזים המנצחים והניקוד של הקבוצה שלהם. רשימת האתגרים מתעדכנת כל מספר שניות.	
מסך בניית קבוצה	רשימה הנשלפת אם השחקן יוצר אתגר משלו. הוא צריך לבחור עונת המשחקים ובה יידרש לבחור שחקנים ששיחקו באותה עונה לנבחרת שלו. אם השחקן נענה לאתגר, הוא לא יוכל לבחור שנה אלא רק להרכיב שחקנים מאותה שנה שבה אותגר. לבחירת שחקן בעמדה מסוימת יש ללחוץ על כפתור העמדה לדוגמה רכז. לאחר הבחירה יוצג שם השחקן שנבחר. על המסך יש תיבת טקסט שבה המשתמש צריך להכניס את שם הקבוצה. בסוף יש כפתור לאישור הנתונים.	מסך זה יכול להיפתח בשני מקרים: אם יש היענות לאתגר, ואם המשתמש רוצה ליצור אתגר משלו. במסך זה המשתמש בונה קבוצה מאותה שנה. המשתמש בוחר שחקנים לפי עמדותיהם במציאות. בנוסף, המשתמש יכניס את שם המאמן, שם הקבוצה ושני שחקנים שירצה לאתגר. בסוף המשתמש ילחץ על כפתור אישור הנתונים.	

מסך רשימת שחקנים	רשימה שחקנים הממוינת לפי שם המשפחה של השחקן.	מסך המכיל רשימת שחקנים מהשנה ומהעמדה שנבחרה. לא ניתן לבחור את אותו שחקן פעמיים. לאחר בחירת שחקן יופיע שם השחקן שנבחר במסך בניית קבוצה.	
מסך רשימת משתמשים	רשימה של משתמשים	מסך המכיל רשימת משתמשים. יש לבחור שני משתמשים, ואסור לבחור אותו משתמש פעמיים. לאחר בחירת משתמש יופיע שמו במסך בניית הקבוצה.	

פירוט המסכים באפליקציה

מסך פתיח



תיאור

מסך זה מוצג למספר שניות ומוצגת התמונה, יש שימוש ב-timer שסופר ומודד את זמן הצגת התמונה.

מקבל

המסך אינו מקבל מידע.

מחזיר

המסך לא מחזיר מידע.

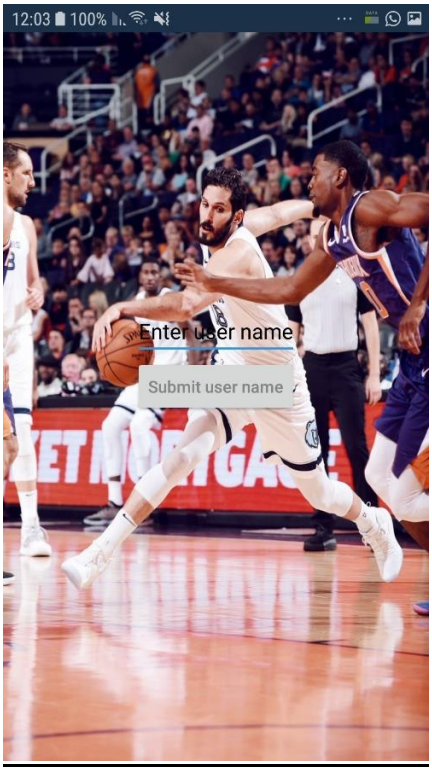
פקדים

אין.

עזר חיצוני

אין.

מסך הרשמה



תיאור

מסך הרשמה חד פעמי שהמשתמש ימלא מיד לאחר התקנת האפליקציה.

המשתמש יכניס את שם המשתמש שלו ויאשר אותו. שם המשתמש יישמר אצלו במכשיר. בכל פעם שהמשתמש נכנס לאפליקציה ייבדק האם שם המשתמש נמצא בשרת, אחרת ייפתח מסך ההרשמה.

מקבל מבחוץ

המסך לא מקבל מידע מבחוץ.

מחזיר למסך הבא

את שם המשתמש.

פקדים

תיבת טקסט להכנסת שם משתמש.

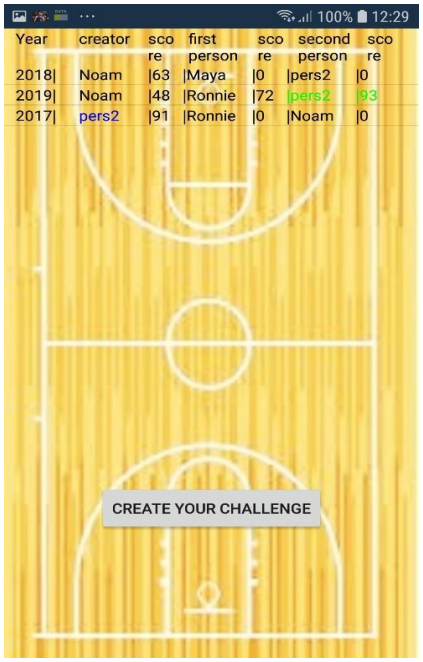
כפתור לאישור הנתונים.

עזר חיצוני

קריאה לשרת לפונקציית ה- API שנקראת user.

שימוש בקובץ מקומי לשמירת שם המשתמש.

מסך האתגרים



תיאור

מסך שבו יצטרך המשתמש לבחור האם להיענות לאתגר שאחרים הציעו לו או ליצור אתגר משלו.
בכחול מסומנים האתגרים שיצרת, בכחול מוכרזים המנצחים והניקוד של הקבוצה שלהם.
רשימת האתגרים מתעדכנת כל מספר שניות.

מקבל מבחוץ

המסך מקבל את שם המשתמש.

מחזיר למסך הבא

אם המשתמש בוחר להיענות לאתגר, מחרוזת הזיהוי של האתגר מועברת למסך בניית קבוצה ועונת המשחקים של האתגר.

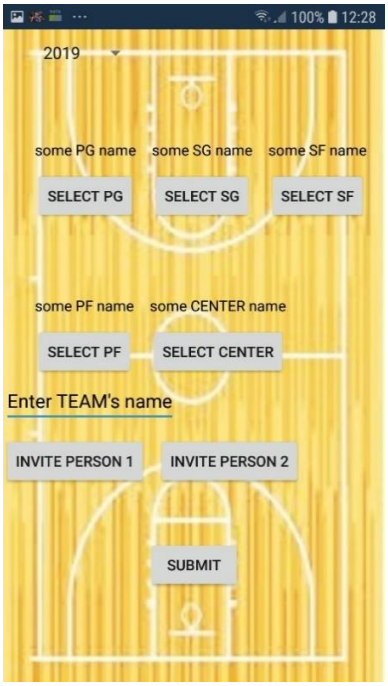
פקדים

רשימה שבה יוצגו כל האתגרים שנשלחו מהמשתמשים האחרים והניקוד של הקבוצות שנבנו.
כפתור של יצירת אתגר חדש.

עזר חיצוני

קריאה לשרת לפונקציית ה -API שנקראת getchals.

מסך בניית קבוצה



תיאור

מסך זה יכול להיפתח בשני מקרים: אם יש היענות לאתגר ואם המשתמש רוצה ליצור אתגר משלו. במסך זה המשתמש בונה קבוצה מאותה שנה. המשתמש בוחר שחקנים לפי עמדותיהם במציאות. בנוסף, המשתמש יכניס את שם המאמן, שם הקבוצה ושני שחקנים שירצה לאתגר. בסוף המשתמש ילחץ על כפתור אישור הנתונים.

מקבל מבחוץ

אם המשתמש בוחר להיענות לאתגר, המסך מקבל ממסך האתגרים את מחרוזת הזיהוי של האתגר, ועונת המשחקים של האתגר. המסך מקבל ממסך השחקנים את המספר המזהה של השחקן שנבחר עבור כל שחקן. אם משתמש יוצר אתגר חדש, המסך מקבל את שם המשתמש שנבחר להזמין לאתגר. המסך מקבל גם את שם המשתמש של המשתמש.

מחזיר למסך הבא

המסך מעביר לרשימת השחקנים את מערך השחקנים שנבחרו, עונת המשחקים ואת העמדה המבוקשת.

פקדים

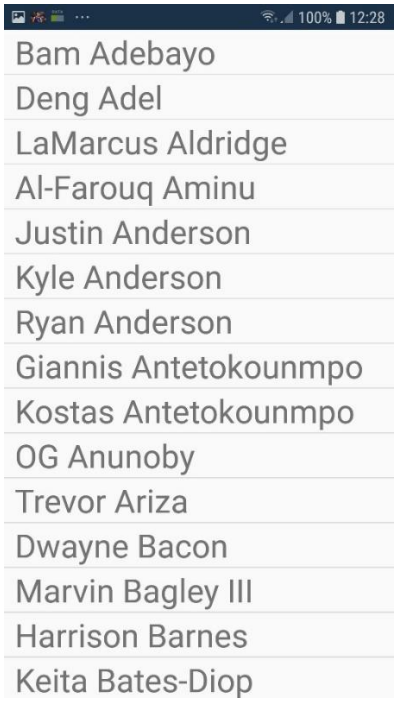
רשימה הנשלפת אם השחקן יוצר אתגר משלו. הוא צריך לבחור עונת משחקים בה יידרש לבחור שחקנים ששיחקו באותה עונה לנבחרת שלו. אם השחקן נענה לאתגר, הוא לא יוכל לבחור שנה אלא רק להרכיב שחקנים מאותה שנה שבה אותגר.

לבחירת שחקן בעמדה מסוימת יש ללחוץ על כפתור העמדה, לדוגמה רכז. לאחר הבחירה יוצג שם השחקן שנבחר. על המסך יש תיבת טקסט שבה המשתמש צריך להכניס את שם הקבוצה. בסוף יש כפתור לאישור הנתונים.

עזר חיצוני

אין עזר חיצוני.

מסך רשימת שחקנים



תיאור

מסך המכיל רשימת שחקנים מהשנה ומהעמדה שנבחרה. לא ניתן לבחור אותו שחקן פעמיים. לאחר בחירת שחקן יופיע שם השחקן שנבחר במסך בניית קבוצה.

מקבל מבחוץ

המסך מקבל ממסך בניית הקבוצה את מערך השחקנים שנבחרו, עונת המשחקים ואת העמדה המבוקשת.

מחזיר למסך הבא

כאשר המשתמש בוחר את השחקן, מוחזר למסך בניית הקבוצה השם המלא של השחקן הנבחר, המספר המזהה שלו והעמדה שלו.

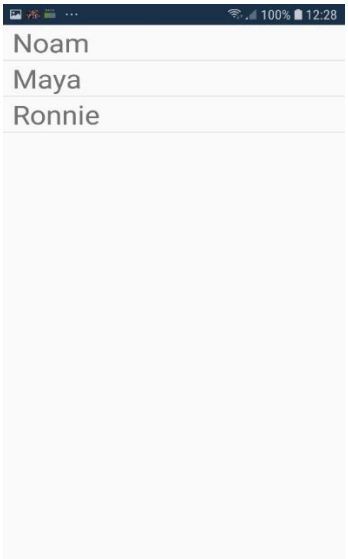
פקדים

רשימה שחקנים הממוינת לפי שם המשפחה של השחקן.

עזר חיצוני

קריאה לשרת לפונקציית ה- API שנקראת retAllPlayersByYear.

מסך רשימת משתמשים



תיאור

מסך המכיל רשימת משתמשים. יש לבחור שני משתמשים, ואסור לבחור אותו משתמש פעמיים. לאחר בחירת משתמש יופיע שמו במסך בניית הקבוצה.

מקבל מבחוץ

המסך מקבל ממסך בניית הקבוצה את מערך השחקנים שנבחרו, עונת המשחקים ואת העמדה המבוקשת.

מחזיר למסך הבא

כאשר המשתמש בוחר את השחקן, מוחזר למסך בניית הקבוצה השם המלא של השחקן הנבחר, המספר המזהה שלו והעמדה שלו.

פקדים

רשימה שחקנים המשתמשים במשחק.

עזר חיצוני

קריאה לשרת לפונקציית ה- API שנקראת getUsers.

Service

1. רכיב - יישום שרץ ברקע ומיועד לבצע פעולות ארוכות
2. service מרחיב את המחלקה הקיימת
3. ה-service רץ ב-thread הראשי

1. startService
    1. יש לו תוחלת חיים משל עצמו.
    2. מוגבל ביכולת שלו לבוא באינטרקציה עם components
    3. רוב ה- services הם started

<u>סדר פעולות יצירת סרויס</u>

ירושה מ- Service

דריסה של onCreate

דריסה של onStartCommand

השימוש ב-Service בפרוייקט

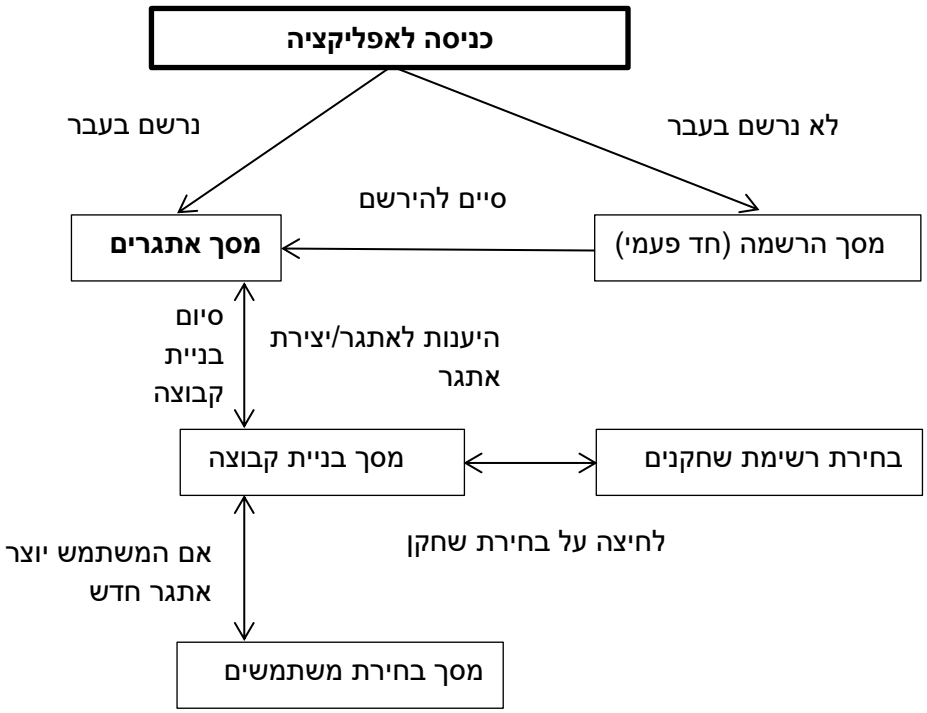
כאשר המשתמש מגיע למסך האתגרים, מתנגן השיר Halleluka ברקע, כך שה-service יורש את הפעולות השימושיות מהמחלקה mediaPlayer שעוזרות לניגון השיר. השיר נשמר בתור קובץ mp3 בפרוייקט ומופיע בשורת המשימות. המשתמש יכול להפעיל ולהפסיק את המוסיקה. בסיום ניגון השיר הוא יתנגן מחדש.

קטע הקוד בפרוייקט המטפל ב- service של ניגון השיר הוא StreamingBackgrondService.

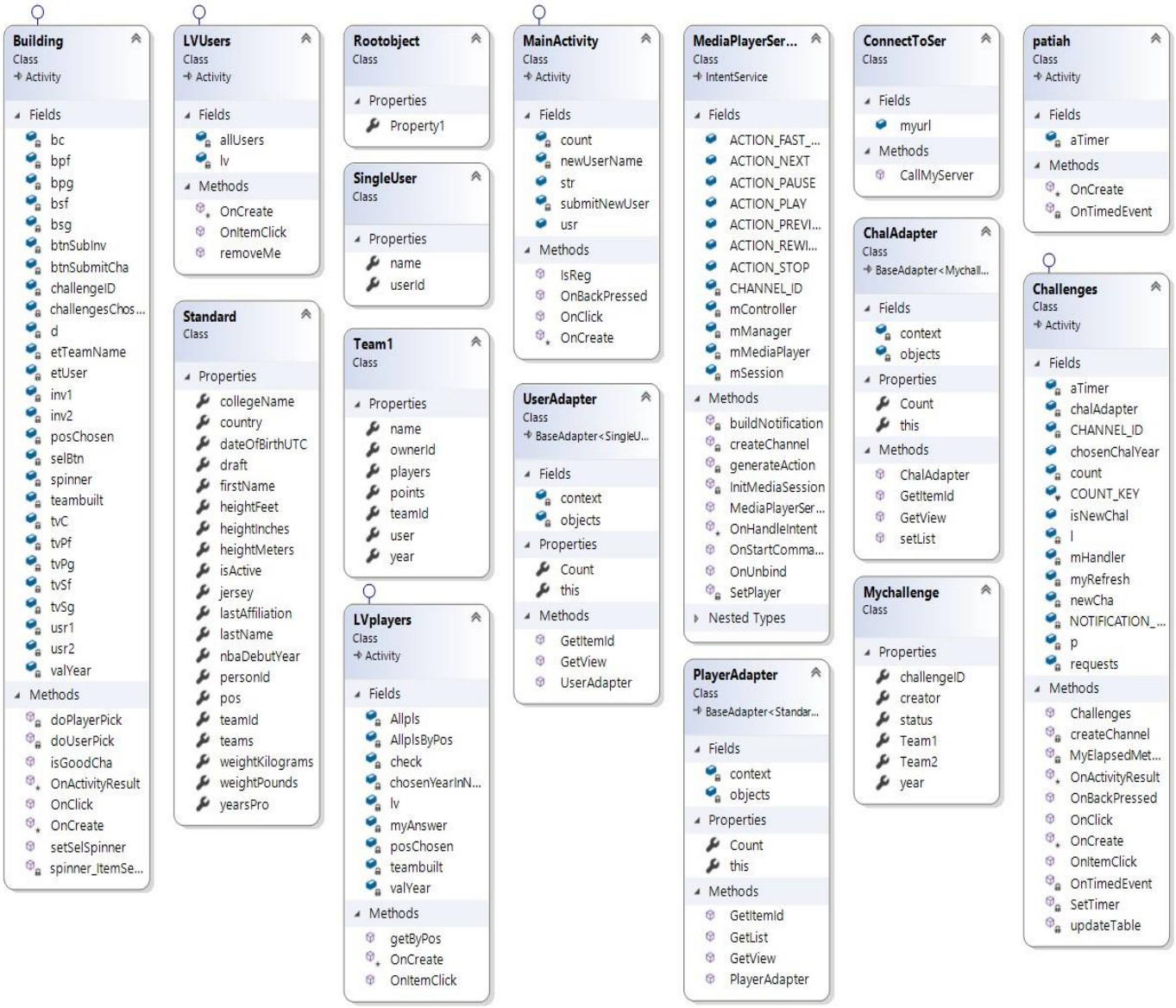
<u>הפעולות המתרחשות בעת ניגון השיר:</u>

- דריסת פעולת play לאחר שיורשים אותה.
- יצירת הודעה על ניגון השיר, והופעת הנגן.
- טיפול באירועי פעולות המשתמש.
- דריסת פעולת עצירת השיר.
- הכנת קובץ השיר לניגון - שליפתו מתוך הזיכרון והכנה שלו.

תרשים זרימה בין המסכים באפליקציה



תרשים UML צד לקוח



מדריך למשתמש

כניסה לאפליקציה

אם התקנת את האפליקציה בפעם הראשונה, יופיע מסך הרשמה. עליך לרשום את שם המשתמש שלך וללחוץ על כפתור אישור. אם נרשמת בעבר, יופיע מסך האתגרים.

לאחר ההרשמה יופיע מסך האתגרים. זהו המסך המרכזי. במסך זה תופיע טבלת האתגרים שנשלחו אליך – מי הוזמן לאתגר, עונת המשחקים שבה ישוחק האתגר, כמה נקודות עשתה כל קבוצה, ומי המנצח אם כולם בנו קבוצה. בנוסף מופיע כפתור של יצירת אתגר חדש.

יצירת אתגר חדש

1. כאשר יוצרים אתגר חדש, עליך לבחור ראשית את עונת המשחקים של האתגר.
2. לאחר מכן יש לבחור שחקנים בכל עמדה.
3. יש ללחוץ על כפתור של הזמנת חבר. לאחר מכן נפתח מסך משתמשים, יש ללחוץ על המשתמש שאותו מזמינים.
4. בתיבת הטקסט יש לרשום את שם המשתמש של החבר שאותו רוצים להזמין ולאשר באמצעות כפתור האישור.
5. לאחר שעשינו כל זאת, יש לאשר את האתגר ולשלוח אותו.
6. לאחר אישור האתגר יופיע מסך האתגרים, וניתן לראות את מצב האתגר במסך האתגרים. ניתן לראות את הסטטוס של כל חבר באתגר, האם הוא סיים לבנות קבוצה.
7. כאשר כל המשתמשים שנענו סיימו לבנות קבוצה, יוצג מסך התוצאות ובו תהיה רשימה של המשתמשים שבנו קבוצה, הניקוד הסופי שלהם ויוכרז המנצח.

היענות לאתגר

1. מתוך רשימת האתגרים שנשלחו לך עליך לבחור אתגר אחד. כאשר בוחרים אתגר שנשלח, יופיע מסך בניית הקבוצה. לא ניתן לבחור שנת אתגרים ולהזמין חברים נוספים לאתגר.
2. עליך לבחור שחקנים ולאשר את הקבוצה שלך.
3. לאחר אישור האתגר יופיע מסך האתגרים, וניתן לראות את מצב האתגר במסך האתגרים. ניתן לראות את הסטטוס של כל חבר באתגר, האם הוא סיים לבנות קבוצה.
4. כאשר כל המשתמשים שנענו סיימו לבנות קבוצה, יוצג במסך האתגרים הניקוד הסופי של כל קבוצה והמנצח.

טבלת פעולות בפייתון

(צד שרת)

GetFromCLi – קובץ זה מטפל בכל הפניות שיש מהלקוח

שם הפעולה	פרמטרים	תפקיד הפעולה
addChal	▪ user- מחרוזת. ▪ pl1,pl2,pl3,pl4,pl5 – מחרוזות. ▪ nameTeam-מחרוזת. ▪ Per1,Per2 – מחרוזות. ▪ year- מחרוזת.	הפעולה מקבלת אתגר חדש תקין המכיל את יוצר האתגר, חמישה id של שחקנים, את שמות המשתמשים של שני המשתמשים שהוזמנו לאתגר ועונת המשחקים. הפעולה יוצרת עצם חדש מטיפוס אתגר. הפעולה מחזירה האם האתגר נוסף בהצלחה.
addTeamToChallenge	▪ idChal- מחרוזת הזיהוי של האתגר בפורמט guid. ▪ pl1, pl2, pl3, pl4, pl5 – מחרוזות. ▪ user- מחרוזת. ▪ teamName – מחרוזת.	הפעולה מקבלת קבוצה תקינה חדשה המכילה שם, חמישה id של שחקנים, שם המשתמש של מי ששלח את הקבוצה ושם הקבוצה. הפעולה תעדכן את ערכי קבוצת המשתמש ששלח את הקבוצה שנוצרה כאשר נוצר האתגר. הפעולה תחזיר שהקבוצה נוספה באופן תקין, אם היא נוספה כצפוי.
user	▪ username – מחרוזת.	הפעולה מקבלת שם משתמש חדש. אם קיים שם משתמש כמו שהתקבל, הפעולה תחזיר שיש לרשום משתמש חדש. אם לא קיים שם משתמש כזה יווצר עצם מטיפוס משתמש עם שם המשתמש החדש שנוצר.
getUsers	הפעולה לא מקבלת פרמטרים	הפעולה מחזירה את רשימת המשתמשים המשחקים במשחק.

retAllPlayersByYear	▪ season- מחרוזת.	הפעולה מקבלת עונת משחקים ומחזירה את רשימת שחקני ה- NBA ששיחקו באותה עונה ומידע בסיסי עליהם.
getchals	▪ user- מחרוזת.	הפעולה מקבלת את שם המשתמש ומחזירה את רשימת האתגרים שהמשתמש נמצא בהם.

מחלקת ממשק שרת API

getFromCLi

(פירוט)

תפקיד מחלקה זו הוא טיפול בפניות הלקוח לצרכים שונים. הפניות ממומשות על ידי הפעולות השונות במחלקה. הפניית הבקשות לפעולות השונות נעשית באמצעות השירות flask.

להלן פירוט הפונקציות:

addChal

תיאור:

הפעולה מוסיפה אתגר חדש, מעדכנת את הקבוצה של יוצר האתגר ויוצרת קבוצות חדשות עבור המשתמשים המוזמנים עם ערכים התחלתיים.

טענת כניסה:

- string -user.
- string – pl1,pl2,pl3,pl4,pl5.
- string - nameTeam.
- string – Per1,Per2.
- string -year.

טענת יציאה:

הפעולה מחזירה האם האתגר נוסף בהצלחה.

דוגמה:

http://127.0.0.1:5000/addChallenge/Noam/1/2/3/4/5/nameTeam/p2/p1/2018

מחזיר:

Added

addTeamToChallenge

תיאור:

הפעולה מקבלת קבוצה תקינה חדשה המכילה שם, חמישה id של שחקנים, שם המשתמש של מי ששלח את הקבוצה ושם הקבוצה. הפעולה תעדכן את ערכי קבוצת המשתמש ששלח את הקבוצה שנוצרה כאשר נוצר האתגר. הפעולה תחזיר שהקבוצה נוספה באופן תקין אם היא נוספה כצפוי.

טענת כניסה:

- idChal- מחרוזת הזיהוי של האתגר בפורמט guid.
- pl1, pl2, pl3, pl4, pl5 – string.
- user- string.
- teamName – string.

טענת יציאה:

הפעולה מחזירה האם הקבוצה נוספה לאתגר בהצלחה.

דוגמה:

http://192.168.3.106:5000/addTeam/97928fb1-5c35-461a-aacb-69bc36bc9513/1626147/1628443/200746/203937/1628387/yy66/pers2

מחזיר:

Added

user

תיאור:

הפעולה מוסיפה משתמשים חדשים. אם קיים שם משתמש כמו שהתקבל, הפעולה תחזיר שיש לרשום משתמש חדש. אם לא קיים שם משתמש כזה, ייווצר עצם מטיפוס משתמש עם שם המשתמש החדש שנוצר.

אם המשתמש מקבל תשובה שקיים משתמש כזה, עליו לשלוח שוב שם משתמש חדש.

טענת כניסה:

- string – username.

טענת יציאה:

הפעולה מחזירה האם המשתמש נוסף בהצלחה, או האם קיים משתמש עם אותו שם.

דוגמה:

http://127.0.0.1:5000/user/OmriTheGOAT

מחזיר:

Added או InList.

getUsers

תיאור:

הפעולה מחזירה את רשימת המשתמשים המשחקים במשחק.

טענת כניסה:

הפעולה לא מקבלת פרמטרים.

טענת יציאה:

רשימת משתמשים בפורמט json מטיפוס users

דוגמה:

http://127.0.0.1:5000/getUsers

מחזיר:

[{"name": "pers2", "userId": "d0c9f72b-79ff-49d6-9b9c-07a908324adc"},
{"name": "Noam", "userId": "5369c492-a242-49f3-98a0-b846cf3f0a85"},
{"name": "Maya", "userId": "679512d7-59d3-4e6b-95f2-b81088d1eb05"},
{"name": "Ronnie", "userId": "44bedade-02e0-485e-ad00-9343bb502b2f"}]

retAllPlayersByYear

תיאור:

הפעולה מחזירה את שחקני ה-NBA ששיחקו או משחקים באותה עונת משחקים.

טענת כניסה:

- string – season.

טענת יציאה:

רשימת שחקנים בפורמט json לפי הפורמט של בסיס הנתונים של ה-NBA שבו אנו משתמשים

דוגמה:

http://127.0.0.1:5000/players/2017

דוגמה של החזרה: דוגמה לשלושה שחקנים ראשונים

{"_internal":{"pubDateTime":"2017-06-23
03:21:52.076","xslt":"xsl/league/roster/marty_active_players.xsl","eventName":"league_roster"},"league":{"standard":[{"firstName":"Alex","lastNam
e":"Abrines","personId":"203518","teamId":"1610612760 1610612760","jersey":"8","pos":"G-
F","heightFeet":"6","heightInches":"6","heightMeters":"1.98","weightPounds":"190","weightKilograms":"86.2","dateOfBirthUTC":"1993-08-
01","teams":[{"teamId":"1610612760","seasonStart":"2016","seasonEnd":"2016"}],"draft":{"teamId":"1610612760","pickNum":"32","roundNum":"2","
seasonYear":"2013"},"nbaDebutYear":"2016","yearsPro":"0","collegeName":"","lastAffiliation":"Spain/Spain","country":"Spain"},{"firstName":"Quinc
y","lastName":"Acy","personId":"203112","teamId":"1610612751
1610612751","jersey":"13","pos":"F","heightFeet":"6","heightInches":"7","heightMeters":"2.01","weightPounds":"240","weightKilograms":"108.9","d
ateOfBirthUTC":"1990-10-
06","teams":[{"teamId":"1610612761","seasonStart":"2012","seasonEnd":"2013"},{"teamId":"1610612758","seasonStart":"2013","seasonEnd":"201
3"},{"teamId":"1610612752","seasonStart":"2014","seasonEnd":"2014"},{"teamId":"1610612758","seasonStart":"2015","seasonEnd":"2015"},{"team
Id":"1610612751","seasonStart":"2016","seasonEnd":"2016"},{"teamId":"1610612742","seasonStart":"2016","seasonEnd":"2016"}],"draft":{"teamId"
:"1610612761","pickNum":"37","roundNum":"2","seasonYear":"2012"},"nbaDebutYear":"2012","yearsPro":"4","collegeName":"Baylor","lastAffiliatio
n":"Baylor/USA","country":"USA"},{"firstName":"Steven","lastName":"Adams","personId":"203500","teamId":"1610612760
1610612760","jersey":"12","pos":"C","heightFeet":"7","heightInches":"0","heightMeters":"2.13","weightPounds":"255","weightKilograms":"115.7","d
ateOfBirthUTC":"1993-07-
20","teams":[{"teamId":"1610612760","seasonStart":"2013","seasonEnd":"2016"}],"draft":{"teamId":"1610612760","pickNum":"12","roundNum":"1","
seasonYear":"2013"},"nbaDebutYear":"2013","yearsPro":"3","collegeName":"Pittsburgh","lastAffiliation":"Pittsburgh/New Zealand","country":"New
Zealand"}]}}

getChals

תיאור:

הפעולה מקבלת את שם המשתמש ומחזירה את רשימת האתגרים שהמשתמש נמצא בהם.

טענת כניסה:

- string -user.

טענת יציאה:

הפעולה מחזירה רשימת עצמים מטיפוס challenge בפורמט json

דוגמה:

http://127.0.0.1:5000/challenges/pers2

מחזיר:

[{"Team1": {"name": "x", "ownerId": null, "players": ["x", "x", "x", "x", "x"], "points": 0, "teamId": "c1577c8c-43a8-46f7-8192-9bb9bacbf90a", "user": "Maya", "year": "2018"}, "Team2": {"name": "x", "ownerId": null, "players": ["x", "x", "x", "x", "x"], "points": 0, "teamId": "2c02f532-b435-4d2e-ab7d-044e1ee7a496", "user": "pers2", "year": "2018"}, "challengeID": "97928fb1-5c35-461a-aacb-69bc36bc9513", "creator": {"name": "xhxhx", "ownerId": null, "players": ["201167", "1628366", "203507", "2772", "1628386"], "points": 63, "teamId": "bedad052-5cd8-47a0-a06b-659a96517c46", "user": "Noam", "year": "2018"}, "status": 0, "year": "2018"}, {"Team1": {"name": "dhdhdh", "ownerId": null, "players": ["1626147", "1628960", "200746", "1628389", "203382"], "points": 72, "teamId": "5aaacb32-bf2b-43fb-8b17-108072679509", "user": "Ronnie", "year": "2019"}, "Team2": {"name": "dhdhd", "ownerId": null, "players": ["201573", "1626147", "201583", "203507", "202326"], "points": 93, "teamId": "8d2f6377-57aa-41dc-9988-0d06eeb554fb", "user": "pers2", "year": "2019"}, "challengeID": "449dd50d-43ed-4b12-b658-510d65154c82", "creator": {"name": "didi", "ownerId": null, "players": ["1627853", "1629121", "1629061", "1627761", "1628436"], "points": 48, "teamId": "c208edee-845f-42ba-a01c-ddb291802732", "user": "Noam", "year": "2019"}, "status": 0, "year": "2019"}, {"Team1": {"name": "x", "ownerId": null, "players": ["x", "x", "x", "x", "x"], "points": 0, "teamId": "4fb81a80-155a-493e-8c17-18330ed95cec", "user": "Ronnie", "year": "2017"}, "Team2": {"name": "x", "ownerId": null, "players": ["x", "x", "x", "x", "x"], "points": 0, "teamId": "5a27c963-4c0c-43d6-9bd3-6ac7c977d8f2", "user": "Noam", "year": "2017"}, "challengeID": "ca625f78-6cd4-4039-b683-ed4065a35801", "creator": {"name": "bbb", "ownerId": null, "players": ["2754", "203937", "101187", "203507", "2199"], "points": 91, "teamId": "317b868b-f7d7-40db-9939-db71a792965f", "user": "pers2", "year": "2017"}, "status": 0, "year": "2017"}]

המשך טבלת פונקציות בפייתון

-sumOfPlayer

שם הפעולה	פרמטרים	תפקיד הפעולה
sumOne	▪ personID – string. ▪ season- string.	הפעולה מקבלת את עונת המשחקים ואת המספר המזהה של השחקן שנקבע ב- API. הפעולה מקבלת את המידע הסטטיסטי של השחקן באמצעות דגימה של בסיס הנתונים, ומחזירה את סכום 5 הפרמטרים הסטטיסטיים של השחקן – נקודות, ריבאונדים, אסיסטים, חטיפות, וחסימות.

-sumOfList

שם הפעולה	פרמטרים	תפקיד הפעולה
sumLst	▪ players – רשימה של מחרוזות. ▪ year- string.	הפעולה מקבלת רשימה של מחרוזות של מספרים מזהים של שחקנים, ועונת משחקים. הפעולה מזמנת את הפעולה sumOne עבור כל איבר ברשימה, ומחזירה את הסכום הסטטיסטי של כל חמשת השחקנים.

getNbaPlayers

המחלקה CachedPlayers

שם הפעולה	פרמטרים	תפקיד הפעולה
getCachedPlayesBySeason	▪ season- string.	פעולה פנימית במחלקה. הפעולה מקבלת עונת משחקים ובודקת האם השרת דגם מידע על שחקנים מהעונה שהתקבלה כפרמטר. אם השרת דגם באותו יום מידע על שחקנים מאותו עונה, מוחזר המידע שנדגם לפני כן באותו יום. אם לא, השרת דוגם את המידע על השחקנים מהעונה המבוקשת, שומר אותו ומחזיר אותו.

CallRest

המחלקה NBA

שם הפעולה	פרמטרים	תפקיד הפעולה
getPlayers	▪ season- string.	פעולה פנימית במחלקה. הפעולה מקבלת עונת משחקים, דוגמת את השרת של ה-NBA ומחזירה את רשימת השחקנים מאותה עונה בפורמט json.

getStats

שם הפעולה	פרמטרים	תפקיד הפעולה
stats	▪ season- string. ▪ idPer – string.	הפעולה מקבלת עונת משחקים ומספר מזהה של שחקן, הפעולה דוגמת את השרת של ה-NBA ומחזירה את הפירוט הסטטיסטי של השחקן המבוקש.

manageTeams

שם הפעולה	פרמטרים	תפקיד הפעולה
__init__	▪ pl1, pl2, pl3, pl4, pl5 – string. ▪ user- string. ▪ teamName – string. ▪ year - string	הפעולה הבונה של המחלקה Team. מקבלת את הפרמטרים ויוצרת עצם מטיפוס קבוצה חדשה.
addTeam	▪ pl1, pl2, pl3, pl4, pl5 – string. ▪ user- string. ▪ teamName – string. ▪ year - string	פעולה בתוך המחלקה MyTeamsClass. הפעולה משתמשת בפעולה הבונה, יוצרת קבוצה, מוסיפה אותה לרשימת הקבוצות וכותבת מחדש את הרשימה לדיסק.
readFile	הפעולה לא מקבלת פרמטרים	פעולה בתוך המחלקה MyTeamsClass. הפעולה שולפת את רשימת הקבוצות מהזיכרון ומחזירה אותה.
writeFile	הפעולה לא מקבלת פרמטרים	הפעולה כותבת מחדש את רשימת הקבוצות לקובץ.

manageChallenges

שם הפעולה	פרמטרים	תפקיד הפעולה
__init__	▪ year – string. ▪ team1 - Team ▪ team2 - Team ▪ tInventor - Team	הפעולה הבונה של המחלקה Challenge. מקבלת את הפרמטרים ויוצרת עצם מטיפוס קבוצה חדשה.
addChallenge	▪ year – string. ▪ team1 - Team ▪ team2 - Team ▪ tInventor - Team	פעולה בתוך המחלקה MyChallenge. הפעולה משתמשת בפעולה הבונה, יוצרת אתגר חדש, מוסיפה אותה לרשימת האתגרים וכותבת מחדש את הרשימה לדיסק.
readFile	הפעולה לא מקבלת פרמטרים	פעולה בתוך המחלקה MyChallenge. הפעולה שולפת את רשימת האתגרים מהזיכרון ומחזירה אותה.
writeFile	הפעולה לא מקבלת פרמטרים	הפעולה כותבת מחדש את רשימת האתגרים לקובץ.
getMyChals	▪ user - string	פעולה בתוך המחלקה MyChallenge. הפעולה עוברת על רשימת האתגרים ומחזירה את רשימת האתגרים שהמשתמש משתתף בהם.
getChalById	▪ chalId	פעולה בתוך המחלקה MyChallenge. הפעולה עוברת על רשימת האתגרים ומחזירה את האתגר שהמזהה שלו כמו שהתקבל.

manageUsers

שם הפעולה	פרמטרים	תפקיד הפעולה
__init__	▪ string – name.	הפעולה הבונה של המחלקה User. מקבלת את שם המשתמש ויוצרת עצם מטיפוס משתמש.
addUser	▪ string – name.	פעולה בתוך המחלקה Users. הפעולה משתמשת בפעולה הבונה, יוצרת משתמש חדש, מוסיפה אותה לרשימת המשתמשים וכותבת מחדש את הרשימה לדיסק.
readFile	הפעולה לא מקבלת פרמטרים	פעולה בתוך המחלקה Users. הפעולה שולפת את רשימת המשתמשים מהזיכרון ומחזירה אותה.
writeFile	הפעולה לא מקבלת פרמטרים	הפעולה כותבת מחדש את רשימת המשתמשים לקובץ.
isExists	▪ string – name.	לפני הוספת המשתמש, הפעולה בודקת האם קיים כבר משתמש עם אותו שם. אם קיים, נשלחת הודעה בחזרה ללקוח שיכתוב שם משתמש חדש. אם לא קיים, יתווסף המשתמש החדש.
userToId	▪ string – user.	פעולה עזר של המחלקה Users. הפעולה מקבלת שם משתמש, עוברת על רשימת המשתמשים ומחזירה את מחרוזת הזיהוי שלו.

מבנה נתונים - צד השרת

לצורך ניהול המידע במשחק הייתי צריך לשמור את הנתונים במבנה נתונים המחולק למספר טבלאות, אותם מימשתי ע"י סידור במחלקות.

לכל מחלקה סיפקתי פעולות ומחלקות עזר (ראה טבלת פעולות בשרת) על מנת לנהל את הנתונים בצורה נוחה (לדוגמה: addChallenge).

על מנת לשמור את המידע השתמשתי בשירות pickle על מנת לסרייל (הפיכת מחלקה למחרוזת) כדי שיהיה אפשר לשמור אותה ולקרוא מהדיסק.

קיימות 3 טבלאות עיקריות שעוזרות לניהול הנתונים:

8. **User**

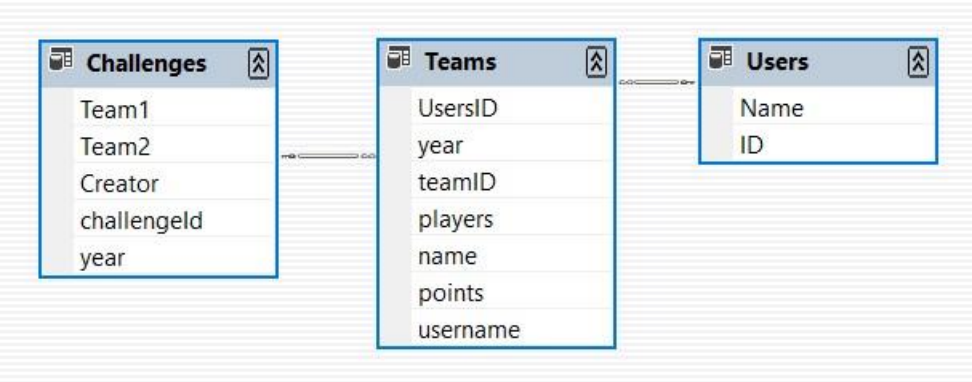
   ניהול המשתמשים במשחק.

9. **Team**

   מבנה הקבוצות במשחק שנוצרות במהלך אתגר.

10. **Challenge**

    ניהול האתגרים הנוצרים במשחק.



המחלקה User

מבנה:

name

- string
- שם המשתמש

userId

- guid
- מזהה ייחודי למשתמש

המחלקה Team

מבנה:

name

- string
- שם הקבוצה

ownerId

- guid
- מזהה ייחודי למשתמש

players

- players = [id1,id2,id3,id4,id5]
- רשימה של מחרוזות של המספר המזהה של השחקן, לפי אתר ה-NBA.

user

- string
- שם המשתמש של בונה הקבוצה

teamId

- guid
- מזהה ייחודי לקבוצה

points

- int
- מספר הנקודות של הקבוצה

year

- string
- עונת המשחקים של הקבוצה באתגר.

המחלקה Challenge

מבנה:

challengeId

- guid
- מזהה ייחודי לאתגר

team1

- מטיפוס Team
- הקבוצה של משתמש מספר 1 שהוזמן לאתגר

team2

- מטיפוס Team
- הקבוצה של משתמש מספר 2 שהוזמן לאתגר

creator

- מטיפוס Team
- הקבוצה של יוצר האתגר

year

- string
- עונת המשחקים של האתגר.

השגת המידע מממשק הנתונים של ה-NBA

על מנת להשיג מידע על ה NBA נדרש לפנות לשרות REST של ה NBA.
בפרויקט זה יש שימוש בשני סוגים של גישות למבנה הנתונים של ה-NBA.

<u>השגת רשימת השחקנים בעונה מסוימת:</u>

כאשר המשתמש בונה קבוצה, הוא צריך לבחור שחקן מרשימת השחקנים. הוא בוחר עמדה, ורשימת השחקנים בעונה המבוקשת מופיעה במסך LVplayers. השרת דוגם את מבנה הנתונים במקרה זה באמצעות REST לפי התבנית הבאה:

http://data.nba.net/10s/prod/v1/{{season}}/players.json
אתר ה-NBA מחזיר את רשימת השחקנים בפורמט Json והשרת מחזיר את המידע באותו אופן.

<u>השגת המידע הסטטיסטי של השחקנים:</u>

הגשת הסטטיסטיקה של שחקן מסוים מתרחשת כאשר קבוצה כלשהי עם שחקנים נוספת לאתגר. השרת סוכם את העמודות הסטטיסטיות של כל השחקנים, ומוחזר הניקוד הכולל של שחקני הקבוצה. השרת משיג את המידע הסטטיסטי של שחקן מסוים בעונת משחקים באופן הבא:

http://data.nba.net/10s/prod/v1{{season}}/players/{{idPer}}_profile.json

סיכום (רפלקציה)

בפרויקט זה למדתי הרבה כלים שימושיים שיכולים לעזור לי מאוד במקצועות המחשב בעתיד. למדתי כיצד לכתוב אפליקציה באנדרואיד – כלי חשוב מאוד ושימושי בעולם שבו הרבה ממה שאנחנו עושים מנוהל במכשיר הסמארטפון שלנו. בפעם הראשונה התנסיתי בכתיבה של פרויקט עם מספר גדול של מחלקות וכיצד להעביר ביניהן מידע בצורה אמינה ונכונה וכיצד לתמרן ולחלק את קטעי הקוד בין המחלקות והמסכים השונים.

בנוסף למדתי כיצד לקשר את האפליקציה לשרת, כלי שיכול לעזור מאוד אם רוצים לקשר בין אנשים ולשמור את המידע מכל מקום בעולם. הקישור בין השרת ללקוח גרם לי לחשוב כיצד אני פועל ומתווך בין שני הצדדים כדי שהתקשורת תהיה ברורה ונכונה.

בעבודה התנסיתי בפעם הראשונה גם כיצד לשמור נתונים מהמשתמש ולחשוב באיזה מבנה נתונים ובאיזה פורמט לשמור את המידע.

גם כתיבת תיק הפרויקט נחוצה למקצועות המחשב בעתיד כי העבודות והפרויקטים שאעשה בעתיד יוגשו ויאורגנו באותה צורה.

הייתי רוצה להודות לגיא שעזר מאוד לקיום הפרויקט והתממשותו, על שהיה עבורי יועץ אסטרטגי ופותר בעיות. תודה רבה על ההנחיה שעזרה לי מאוד עם ההתנהלות של העבודה, ולמדתי לחשוב בצורה נכונה, מסודרת, פשוטה ואלגנטית. לדעתי זהו כלי חשוב מאוד שיעזור לי בעתיד בתור מתכנת. נעזרתי גם באסף אמיר שעזר ולימד בצורה ברורה מאוד את הנושאים השונים באנדרואיד.

קטע קוד מלא

צד השרת

getFromCLi.py

```
# -*- coding: utf-8 -*-

from flask import Flask
import getstats
import sumOfPlayer
from flask import Flask
import tstLogin
import manageUsers
import manageTeams
import jsonpickle
import manageChallenge
from flask import jsonify
import manageChallenge
from manageChallenge import Challenge
import json
import sumOfList

import CallRest
from manageChallenge import MyChallenge
import getNbaPlayers


app = Flask(__name__)



lst = []
globalMyChallenge = MyChallenge()

"""@app.route("/")
def hello():
    return "Hello World!"


@app.route("/guy")
def helloGuy():
    return "Hello Guy!"""""

@app.route("/addChallenge/<user>/<pl1>/<pl2>/<pl3>/<pl4>/<pl5>/<nameTeam>/<Per1>/<Per2>/<y
ear>")
def addChal(user, pl1,pl2,pl3,pl4,pl5,nameTeam,Per1,Per2,year):


    p = manageTeams.MyTeamsClass()

    inventor, y = p.addTeam(pl1, pl2, pl3, pl4, pl5, user, nameTeam, year)
    chal, status = globalMyChallenge.addNewChallenge(year, inventor, Per1, Per2)
    x = globalMyChallenge.getMyChals(Per1)
    x = globalMyChallenge.toJSON(x)
    return status
```

```
"""
@app.route('/getsum/<id>/<season>')
def show_sum(id,season):
    print ("id = " + id + " season = " + season)
    return str(sumOfPlayer.sumOne(id,season))
"""
@app.route('/addTeam/<idChal>/<pl1>/<pl2>/<pl3>/<pl4>/<pl5>/<teamName>/<user>')
def addTeamToChallenge(idChal, pl1, pl2, pl3, pl4, pl5, teamName, user):


    c = globalMyChallenge.getChalById(idChal)

    print("addTeamToChallenge, year = " + c.Team1.year)

    if c.Team1.user == user:
        c.Team1.players[0] = pl1
        c.Team1.players[1] = pl2
        c.Team1.players[2] = pl3
        c.Team1.players[3] = pl4
        c.Team1.players[4] = pl5
        c.Team1.name = teamName
        c.Team1.points = sumOfList.sumlst([pl1,pl2,pl3,pl4,pl5], c.Team1.year)
    else:
        c.Team2.players[0] = pl1
        c.Team2.players[1] = pl2
        c.Team2.players[2] = pl3
        c.Team2.players[3] = pl4
        c.Team2.players[4] = pl5
        c.Team2.name = teamName
        c.Team2.points = sumOfList.sumlst([pl1,pl2,pl3,pl4,pl5], c.Team2.year)
    t = manageTeams.MyTeamsClass()
    t.writeFile()
    globalMyChallenge.writeFile()
    return "Added"


@app.route('/user/<username>')
def show_user_profile(username):
    # show the user profile for that user
    #TestList.Users.addUser()

    v = manageUsers.Users()
    return v.addUser(username)

@app.route('/getUsers')
def getUsers():
    # returns all users in the game
    v = manageUsers.Users()

    y = v.getUsers()
    print(y)

    print("x = " +str(y))
    x = v.toJSON(y)

    m = jsonpickle.encode(y, unpicklable=False)
    print(m)
    return m
```

```
@app.route('/players/<season>')
def retAllPlayersByYear(season):

    x = getNbaPlayers.CachedPlayers()
    result = x.getCachedPlayesBySeason(season)

    return result

    p = CallRest.Nba()
    t  = p.test(season)
    # print ("players are=" + str(y))

    xx = str(y)
    # j = json.dumps(xx)
    # print ("xx = " + xx )
    # s = str(j)
    print ("t = " + t)
    return t


@app.route('/challenges/<user>')
def getchals(user):
    # show the user profile for txxxhat user
    v = MyChallenge()
    #y = globalMyChallenge.getMyChals(user)
    y = v.getMyChals(user)
    print (y)

    print("x = " +str(y))
    x = v.toJSON(y)

    m = jsonpickle.encode(y, unpicklable=False)
    print(m)
    return m




"""
@app.route('/Team/<id1>/<id2>/<id3>/<id4>/<id5>/<ownerID>/<name>/<year>')
def show_Teams_profile(id1,id2,id3,id4,id5, ownerID, name, year):
    return manageTeams.addUser(id1,id2,id3,id4,id5, ownerID, name)



@app.route('/Team/<id1>/<id2>/<id3>/<id4>/<id5>/<ownerID>/<name>')
def show_Teams_profile2(id1,id2,id3,id4,id5, ownerID, name, year):
    return manageTeams.addUser(id1,id2,id3,id4,id5, ownerID, name)



@app.route('/add/<int:a>/<int:b>')
def addme(a, b):
    # show the user profile for that user
    print ("a = " + str(a) + " b = " + str(b))
    x = a + b
    v = str(x)

    lst.append(v)
```

```
    print (lst)
    noam = str(lst)
    f = open( 'c:\\temp\\users.txt', 'w')
    f.write(noam)
    f.close()

    return v

"""


if __name__ == '__main__':

    globalMyChallenge.readFile()

    #test = globalMyChallenge.getMyChals("Noam")


    app.run(host="0.0.0.0")
```

CallRest.py

```
# -*- coding: utf-8 -*-
import requests
import json
from flask import jsonify

def main():
    """
    Add Documentation here
    """
    p = Nba()
    x = p.test("2018")
    print (x)
class Nba:
    def test(self, year):

        y = str(int(year)-1)

        r = 'http://data.nba.net/10s/prod/v1/' + y+"/players.json"
        req = requests.get(r)
        p = req.text

        data = json.loads(req.text)

        return req.text

    #print data

    lst = []


if __name__ == '__main__':
    main()
```

getNBAPlayers.py

```
import CallRest
import datetime

class Players(object):
    """__init__() functions as the class constructor"""
    def __init__(self, season, updateTime, result):
        self.season = season
        self.updateTime = updateTime
        self.result = result


class CachedPlayers:
    myCachedPlayers = []



    def getCachedPlayesBySeason(self,season):

        t = CachedPlayers.myCachedPlayers

        for i in t:
            if i.season == season:
                deltaTime = datetime.datetime.now() - i.updateTime
                if deltaTime.days == 0:
                    return i.result
                else:
                    p = CallRest.Nba()
                    t = p.test(season)
                    i.result = t
                    i.updateTime = datetime.datetime.now()
                    return i.result


        p = CallRest.Nba()
        t = p.test(season)

        players = Players(season, datetime.datetime.now(), t)
        CachedPlayers.myCachedPlayers.append(players)

        return t
```

getStats.py

```
import requests
import json
import pickle

def main():
    """
    Add Documentation here
    """
    stats("977","2015")


def stats(idPer,season):

    try:

        season = str(int(season)-1)

        print("season = " + season)
        r = "http://data.nba.net/10s/prod/v1/"+season+"/players/"+idPer+"_profile.json"
        print("r = " + r)
        req = requests.get(r)
        data = json.loads(req.text)

        return data["league"]["standard"]["stats"]["regularSeason"]["season"][0]["total"]

    except:
        return "no information"


if __name__ == '__main__':
    main()
```

manageChallenge

```
# -*- coding: utf-8 -*-
import flask
import requests
import pickle
import json
import json
import os.path
import pickle
import uuid
from json import JSONEncoder
from json import JSONDecoder
import sumOfList
import manageTeams
import manageUsers
from flask import jsonify

class Challenge:
    """__init__() functions as the class constructor"""
    def __init__(self, year, t1, t2, tInventor):

        self.status = 0
        self.challengeID = str(uuid.uuid4())
        self.creator = tInventor
        self.Team1 = t1
        self.Team2 = t2
        self.year = year


class MyChallenge:


    myChallenges = []

    def toJSON(self, onlymine):
        return json.dumps(onlymine, default=lambda o: o.__dict__,
            sort_keys=True, indent=4)

    def addChallenge(self,year, t1, t2, tInventor):

        newC = Challenge(year, t1, t2, tInventor)
        MyChallenge.myChallenges.append(newC)
        self.writeFile()
        return newC, "Added"

    def addNewChallenge(self, year, tInventor, USR1, USR2):

        p = manageTeams.MyTeamsClass()
        t1, y = p.addTeam("x","x","x","x","x",USR1,"x",year)
        t2, y2 = p.addTeam("x", "x", "x", "x", "x", USR2, "x", year)

        newC = Challenge(year, t1, t2, tInventor)
        MyChallenge.myChallenges.append(newC)
        self.writeFile()
        return newC, "Added"

    def isExist(self, nameTeam):
        for i in MyChallenge.myChallenges:
```

```
            if i.Team2.players[0] is "1":
                return True
                break

        return False


    def readFile(self):
        p = "C:\\tstJson\\Challenges.json"

        #global myChallenges
        if os.path.exists(p):
            m = open(p, "rb")
            if os.path.getsize(p) > 0:
                MyChallenge.myChallenges = pickle.load(m)
            else:
                MyChallenge.myChallenges = []
            m.close()
        else:
            print("not found")


    def getMyChals(self, userPar):
        lst = []

        t = MyChallenge.myChallenges
        for i in t:

            if i.Team2.user == userPar:
                lst.append(i)
            if i.Team1.user == userPar:
                lst.append(i)
            if i.creator.user == userPar:
                lst.append(i)
        return lst


    def writeFile(self):
        x = pickle.dumps(MyChallenge.myChallenges)
        p = "C:\\tstJson\\Challenges.json"
        g = open(p, "wb")
        g.write(x)
        g.close()

    def getChalById(self,chalId):
            for i in MyChallenge.myChallenges:
                if i.challengeID == chalId:
                    return i
            return


def main():
    """
    Add Documentation here
    """
    pass  # Add Your Code Here


if __name__ == '__main__':
    main()
```

manageTeams.py

```
import flask
import requests
import pickle
import json
import json
import os.path
import pickle
import uuid
from json import JSONEncoder
from json import JSONDecoder
import manageUsers
import sumOfList

class Team(object):
    """__init__() functions as the class constructor"""
    def __init__(self, id1,id2,id3,id4,id5, user, nameTeam, year):
        self.players = [id1,id2,id3,id4,id5]
        self.name = nameTeam

        self.user = user

        v = manageUsers.Users()
        userId = v.userToID(user)

        self.ownerId = userId
        self.year = year
        self.teamId = str(uuid.uuid4())
        if id1 != "x":
            self.points = sumOfList.sumlst([id1,id2,id3,id4,id5],year)
        else:
            self.points = 0

class MyTeamsClass:
    myTeamsList = []


    def addTeam(self, id1,id2,id3,id4,id5, user, nameTeam, year):

        newT = Team(id1,id2,id3,id4,id5, user, nameTeam, year)
        self.myTeamsList.append(newT)
        self.writeFile()
        return newT, "Added"



    def retTheT(self, name):
        for i in self.myTeamsList:
            if i.name == name:
                return i
                break



    def isExist(self, nameTeam):
        for i in self.myTeamsList:
            if i.name == nameTeam:
                return True
```

```
                break

        return False

    #def printMyList():
    #    #print myUsers[0]
    #    for i in myTeams:
    #        print (i.name + ", id = " + i.ownerId)


    def readFile(self):
        p = "C:\\tstJson\\Teams.json"

#        global myTeams
        if os.path.exists(p):
            m = open(p, "rb")
            if os.path.getsize(p) > 0:
                self.myTeamsList = pickle.load(m)
            else:
                self.myTeamsList = []
            m.close()
        else:
            print("not found")

    def writeFile(self):

        x = pickle.dumps(self.myTeamsList)
        p = "C:\\tstJson\\Teams.json"
        g = open(p, "wb")
        g.write(x)
        g.close()




def main():
    """
    Add Documentation here
    """
    #printMyList()


if __name__ == '__main__':
    main()
```

manageUsers.py

```
# -*- coding: utf-8 -*-
from json import JSONEncoder
from json import JSONDecoder
import json
import os.path
import pickle
import uuid



class User(object):
    """__init__() functions as the class constructor"""
    def __init__(self, name=None):

        self.name = name
        self.userId = str(uuid.uuid4())


class Users:
    myUsers = []

    def __init__(self):
        self.readFile()

    def toJSON(self, onlymine):
        return json.dumps(onlymine, default=lambda o: o.__dict__,
            sort_keys=True, indent=4)

    def getUsers(self):
        return Users.myUsers

    def addUser(self, name):

        if not self.isExist(name):
            Users.myUsers.append(User(name))
            self.writeFile()
            return "Added"
        else:
            return "InList"


    def isExist(self, name):
        for i in Users.myUsers:
            if i.name == name:
                return True
                break

        return False

    #def printMyList():
    #    #print myUsers[0]
    #    for i in myUsers:
    #        print (i.name + ", id = " + i.userId)
```

```
    def readFile(self):
        p = "C:\\tstJson\\users.json"
        if os.path.exists(p):
            m = open("C:\\tstJson\\users.json", "rb")
            if os.path.getsize(p) > 0:

                Users.myUsers = pickle.load(m)
            else:
                Users.myUsers = []
            m.close()
        else:
            Users.myUsers = []

    def writeFile(self):

        x = pickle.dumps(Users.myUsers)
        p = "C:\\tstJson\\users.json"
        g = open(p, "wb")
        g.write(x)
        g.close()





        #f = open(p, "wb")
        #f.write(content)
        #f.close()


    def userToID(self, user):
        for i in Users.myUsers:
            if i.name is user:
                return i.userId


    def idToName(self, idPar):
        for i in Users.myUsers:
            if i.userId is idPar:
                return i.name



def main():
    """
    Add Documentation here
    """
#    pass  # Add Your Code Here
#    myUsers.append(addUser("a",23))
    readFile()

    print ("add = " + addUser("Noam", 33))
    print ("add = " + addUser("WOW", 22))
    print ("add = " + addUser("Ronnie", 44))
    print ("add = " + addUser("Ronnie", 44))

    printMyList()

if __name__ == '__main__':
    main()
```

sumOfList.py

```
import sumOfPlayer


def main():
    """
    Add Documentation here
    """

    sumlst(["201933", "977", "201933"], "2015")

    pass  # Add Your Code Here


def sumlst(players, year):

    sumplayers = 0

    for i in players:
        sumplayers += sumOfPlayer.sumOne(i,year)

    return sumplayers


if __name__ == '__main__':
    main()
```

sumOfPlayer.py

```
import requests
import json
import math
import getstats

def main():
    """
    Add Documentation here
    """

    pass

def sumOne(personID,season):

    stats = getstats.stats(personID,season)
    if stats == "no information":
        print ("0 because no information")
        return 0
    ppg = float(stats["ppg"])
    if ppg == -1:
        print("0 because didnt play")
        return 0
    apg = float(stats["apg"])
    rpg = float(stats["rpg"])
    spg = float(stats["spg"])
    bpg = float(stats["bpg"])

    tot = int(ppg+apg+rpg+spg+bpg)


    print (tot)
    return tot
```

צד הלקוח

XML

challengesNew.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
       android:background="@drawable/bcourt2"
       >

       <LinearLayout
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:orientation="horizontal"

              >

              <TextView
                     android:layout_width="45dp"
                     android:layout_height="wrap_content"
                     android:text="Year"
                     android:layout_marginLeft="10dp"
                     android:textColor="@android:color/black"
              />


              <TextView
                     android:layout_width="50dp"
                     android:layout_height="wrap_content"
                     android:text="creator"
                     android:layout_marginLeft="10dp"
                     android:textColor="@android:color/black"
              />

              <TextView
                     android:layout_width="25dp"
                     android:layout_height="wrap_content"
                     android:text="score"
                     android:layout_marginLeft="10dp"
                     android:textColor="@android:color/black"
              />


              <TextView
                     android:layout_width="50dp"
                     android:layout_height="wrap_content"
                     android:text="first person"
                     android:layout_marginLeft="10dp"
                     android:textColor="@android:color/black"
              />

<!--> 5 <-->
              <TextView
                     android:layout_width="25dp"
                     android:layout_height="wrap_content"
```

```
                        android:text="score"
                        android:layout_marginLeft="10dp"
                        android:textColor="@android:color/black"

                />
<!--> 6 <-->
                <TextView
                        android:layout_width="50dp"
                        android:layout_height="wrap_content"
                        android:text="second person"
                        android:layout_marginLeft="10dp"
                        android:textColor="@android:color/black"
                />
<!--> 7 <-->
                <TextView
                        android:layout_width="25dp"
                        android:layout_height="wrap_content"
                        android:text="score"
                        android:layout_marginLeft="10dp"
                        android:textColor="@android:color/black"
                />



        </LinearLayout>

        <ListView
                android:layout_width="match_parent"
                android:layout_height="400dp"
                android:id="@+id/lvChallenges"

                >
        </ListView>



        <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="create your challenge"
                android:layout_gravity="center"
                android:layout_marginTop="10dp"
                android:id="@+id/btnNewChallenge"
        />


        <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Refresh"
                android:layout_gravity="center"
                android:layout_marginTop="10dp"
                android:id="@+id/btnRefresh"
        />



</LinearLayout>
```

Harshama.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
       xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
       android:background="@drawable/oc_shamg">

       <EditText
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:hint="Enter user name"
              android:layout_gravity="center"
              android:layout_marginTop="230dp"
              android:id="@+id/etNewUname"
              android:textColorHint="@android:color/black"

       />

       <Button
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:hint="Submit user name"
              android:layout_gravity="center"
              android:id="@+id/btnNewUser"

       />


</LinearLayout>
```

listPLayers.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

       <ListView
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:id="@+id/lv"
       />


</LinearLayout>
```

listUsers.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

       <ListView
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:id="@+id/lvUsers"
       />


</LinearLayout>
```

newTeam.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
       android:background="@drawable/bcourt2">

        <Spinner
        android:id="@+id/spinner"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:prompt="@string/action_settings"
              android:layout_marginTop="10dp"
              android:layout_marginLeft="30dp"
    />

       <LinearLayout
              android:layout_marginTop="60dp"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:orientation="horizontal"

       >
              <LinearLayout
                     android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_marginLeft="30dp"
>

        <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="some PG name"
                android:layout_marginTop="10dp"
                android:id="@+id/tvpg"
                android:textColor="@android:color/black"

        />

        <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="select PG"
                android:layout_marginTop="10dp"
                android:id="@+id/btnpg"

        />


</LinearLayout>

<LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_marginLeft="15dp"
>

        <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="some SG name"
                android:layout_marginTop="10dp"
                android:id="@+id/tvsg"
                android:textColor="@android:color/black"
        />

        <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="select SG"
                android:layout_marginTop="10dp"
                android:id="@+id/btnsg"

        />


</LinearLayout>

<LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_marginLeft="15dp"
>
```

```
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="some SF name"
                android:layout_marginTop="10dp"
                android:id="@+id/tvsf"
                android:textColor="@android:color/black"
            />

            <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="select SF"
                android:layout_marginTop="10dp"
                android:id="@+id/btnsf"

            />


        </LinearLayout>




</LinearLayout>


        <LinearLayout
        android:layout_marginTop="60dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"

>
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:layout_marginLeft="30dp"
        >

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="some PF name"
                android:layout_marginTop="10dp"
                android:id="@+id/tvpf"
                android:textColor="@android:color/black"
            />

            <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="select PF"
                android:layout_marginTop="10dp"
                android:id="@+id/btnpf"

            />


        </LinearLayout>
```

```
        <LinearLayout
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:orientation="vertical"
                android:layout_marginLeft="15dp"
        >

                <TextView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="some CENTER name"
                        android:layout_marginTop="10dp"
                        android:id="@+id/tvc"
                        android:textColor="@android:color/black"
                />

                <Button
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="select center"
                        android:layout_marginTop="10dp"
                        android:id="@+id/btncenter"

                />


        </LinearLayout>


</LinearLayout>



<EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="Enter TEAM's name"
        android:id="@+id/etNewTeamName"
        android:textColorHint="@android:color/black"

/>

        <LinearLayout
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:orientation="horizontal"
        >

        <Button

                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="invite person 1"
                android:layout_marginTop="10dp"
                android:id="@+id/btnInvite1"


        />

        <Button
```

```
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="invite person 2"
                android:layout_marginTop="10dp"
                android:layout_marginLeft="10dp"
                android:id="@+id/btnInvite2"


            />


            </LinearLayout>




        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="submit"
            android:layout_gravity="center"
            android:layout_marginTop="50dp"
            android:id="@+id/btnCreateCha"
        />


</LinearLayout>
```

oneItemChallenge.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal"

            >

            <TextView
                android:layout_width="45dp"
                android:layout_height="wrap_content"
                android:text="on year"
                android:layout_marginLeft="10dp"
                android:id="@+id/tvTypeChallenge"
                android:textColor="@android:color/black"
            />

            <TextView
                android:layout_width="50dp"
                android:layout_height="wrap_content"
                android:text="challenges you!"
                android:layout_marginLeft="10dp"
                android:id="@+id/tvUser"
                android:textColor="@android:color/black"
```

```
            />

            <TextView
                    android:layout_width="20dp"
                    android:layout_height="wrap_content"
                    android:text="score"
                    android:layout_marginLeft="10dp"
                    android:id="@+id/tvScoreCreator"
                    android:textColor="@android:color/black"
            />

            <TextView
                    android:layout_width="50dp"
                    android:layout_height="wrap_content"
                    android:text="first person"
                    android:layout_marginLeft="10dp"
                    android:id="@+id/tvPer1"
                    android:textColor="@android:color/black"
            />

<!--> 5 <-->
            <TextView
                    android:layout_width="20dp"
                    android:layout_height="wrap_content"
                    android:text="score"
                    android:layout_marginLeft="10dp"
                    android:id="@+id/tvScorePer1"
                    android:textColor="@android:color/black"
            />

            <TextView
                    android:layout_width="50dp"
                    android:layout_height="wrap_content"
                    android:text="second person"
                    android:layout_marginLeft="10dp"
                    android:id="@+id/tvPer2"
                    android:textColor="@android:color/black"
            />

            <TextView
                    android:layout_width="20dp"
                    android:layout_height="wrap_content"
                    android:text="score"
                    android:layout_marginLeft="10dp"
                    android:id="@+id/tvScorePer2"
                    android:textColor="@android:color/black"
            />


        </LinearLayout>


</LinearLayout>
```

oneItemPlayer.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

       <TextView
              android:id="@+id/tvFull"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:hint="Full name"
              android:textSize="30sp"
              android:layout_marginLeft = "10dp"
       />

</LinearLayout>
```

oneUser.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

       <TextView
              android:id="@+id/tvFull"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:hint="Full name"
              android:textSize="30sp"
              android:layout_marginLeft = "10dp"
       />

</LinearLayout>
```

Patiah.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
       android:background="@drawable/patiah">

</LinearLayout>
```

C#

Building.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

namespace My_Fantasy
{
    [Activity(Label = "Building")]

    public class Building : Activity, Android.Views.View.IOnClickListener
    {
        int selBtn = 0;
        Button btnSubmitCha;

        Button inv1;
        Button inv2;
        Dialog d;
        Button btnSubInv;
        EditText etUser;
        static Spinner spinner;
        Button bpg, bsg, bsf, bpf, bc;
        static string posChosen = "";
        static string valYear = "2019";
        static String[] teambuilt;
        static TextView tvPg, tvSg, tvSf, tvPf, tvC;
        static EditText etTeamName;
        static string usr1, usr2;
        static string challengesChosenChalYear;
        string challengeID;

        public void OnClick(View v)
        {

            if (v == inv1 || v == inv2)
            {

                //if the user pressed to invite another user

                if(v == inv1)
                {
                    selBtn = 1;
                }
                else
                {
                    selBtn = 2;
                }

                //pick user from list
```

```
    Android.Content.Intent intent = new Android.Content.Intent(this, typeof(LVUsers));
    StartActivityForResult(intent, 2);

}



if (v == bc || v == bpf || v == bsf || v == bsg || v == bpg)
{
    if (v == bc)
    {
        posChosen = "C";

    } else if (v == bpf)
    {
        posChosen = "PF";
    }
    else if (v == bsf)
    {
        posChosen = "SF";
    }
    else if (v == bsg)
    {
        posChosen = "SG";
    }
    else
    {
        posChosen = "PG";
    }

    Android.Content.Intent intent = new Android.Content.Intent(this, typeof(LVplayers));

    intent.PutExtra("valYear", valYear);
    intent.PutExtra("teambuilt", teambuilt);
    intent.PutExtra("posChosen", posChosen);

    StartActivityForResult(intent, 1);

}




if (v == btnSubInv)
{
    string i = etUser.Text;
    //bool b = findPer(i);

    bool b = true;


}

if (v == btnSubmitCha)
{
    if (isGoodCha())
    {
        string joinedArr = string.Join("/", teambuilt);
        if (challengesChosenChalYear == null )
        {
```

```
                //create Challenge

                ConnectToSer c = new ConnectToSer();

                string ans = c.CallMyServer("addChallenge",
MainActivity.usr+"/"+joinedArr+"/"+etTeamName.Text+"/"+usr1+"/"+usr2+"/"+valYear);
                if (ans.Equals("Added"))
                {


                    Intent intent = new Intent();
                    SetResult(Result.Ok, intent);
                    Finish();
                    //send some vars with intent

                }

            }
            else
            {
                //send team to server and edit challenge
                ConnectToSer c = new ConnectToSer();
                string ans = c.CallMyServer("addTeam", challengeID+"/"+joinedArr+"/"+
etTeamName.Text + "/"+MainActivity.usr);
                Intent intent = new Intent();
                SetResult(Result.Ok, intent);
                Finish();
            }

        }
    }



}



    protected override void OnActivityResult(int requestCode, [GeneratedEnum] Result resultCode,
Intent data)
    {
        base.OnActivityResult(requestCode, resultCode, data);

        if ( resultCode != Result.Ok)
        {
            return;
        }

        switch(requestCode)
        {
            case 1:
                doPlayerPick(data);
                break;
            case 2:
                doUserPick(data);
                break;
        }
    }

    private void doUserPick(Intent data)
```

```
        {
            string returnName = data.GetStringExtra("name");
            string returnUserId = data.GetStringExtra("userId");

            if (!string.IsNullOrEmpty(usr1) && usr1.Equals(returnName))
            {
                Toast.MakeText(this, "you have already chose user "+returnName,
ToastLength.Long).Show();
                return;
            }

            if (!string.IsNullOrEmpty(usr2) && usr2.Equals(returnName))
            {
                Toast.MakeText(this, "you have already chose user " + returnName,
ToastLength.Long).Show();
                return;
            }

            //etUser.Text = returnName;

            if (selBtn == 1)
            {
                usr1 = returnName;
                inv1.Text = usr1;
            }
            else
            {
                usr2 = returnName;
                inv2.Text = usr2;
            }

        }

        private void doPlayerPick(Intent data)
        {
            string returnPlayerId = data.GetStringExtra("playerId");
            string returnPlayerFullName = data.GetStringExtra("playerFullName");
            string returnPosChosen = data.GetStringExtra("posChosen");

            Toast.MakeText(this, "returnPlayerFullName = " + returnPlayerFullName,
ToastLength.Long).Show();
            spinner.Enabled = false;



            switch (returnPosChosen)
            {
                case "PG":
                    tvPg.Text = returnPlayerFullName;
                    teambuilt[0] = returnPlayerId;
                    break;
                case "SG":
                    tvSg.Text = returnPlayerFullName;
                    teambuilt[1] = returnPlayerId;
                    break;
                case "SF":
                    tvSf.Text = returnPlayerFullName;
                    teambuilt[2] = returnPlayerId;
                    break;
                case "PF":
```

```
                tvPf.Text = returnPlayerFullName;
                teambuilt[3] = returnPlayerId;
                break;
            case "C":
                tvC.Text = returnPlayerFullName;
                teambuilt[4] = returnPlayerId;
                break;
        }

    }

    protected override void OnCreate(Bundle savedInstanceState)
    {


        base.OnCreate(savedInstanceState);
        SetContentView(Resource.Layout.newTeam);

        challengesChosenChalYear = Intent.GetStringExtra("chosenChalYear");
        challengeID = Intent.GetStringExtra("challengeID");
        //challengesIsNewChal = Intent.GetBooleanExtra("isNewChal",false);


        spinner = FindViewById<Spinner>(Resource.Id.spinner);

        spinner.ItemSelected += new
EventHandler<AdapterView.ItemSelectedEventArgs>(spinner_ItemSelected);
        var adapter = ArrayAdapter.CreateFromResource(
                this, Resource.Array.years_array, Android.Resource.Layout.SimpleSpinnerItem);




adapter.SetDropDownViewResource(Android.Resource.Layout.SimpleSpinnerDropDownItem);
        spinner.Adapter = adapter;

        tvC = FindViewById<TextView>(Resource.Id.tvc);
        tvPg = FindViewById<TextView>(Resource.Id.tvpg);
        tvSg = FindViewById<TextView>(Resource.Id.tvsg);
        tvPf = FindViewById<TextView>(Resource.Id.tvpf);
        tvSf = FindViewById<TextView>(Resource.Id.tvsf);



        bpg = FindViewById<Button>(Resource.Id.btnpg);
        bpg.SetOnClickListener(this);

        bsg = FindViewById<Button>(Resource.Id.btnsg);
        bsg.SetOnClickListener(this);

        bsf = FindViewById<Button>(Resource.Id.btnsf);
        bsf.SetOnClickListener(this);

        bpf = FindViewById<Button>(Resource.Id.btnpf);
        bpf.SetOnClickListener(this);

        bc = FindViewById<Button>(Resource.Id.btncenter);
        bc.SetOnClickListener(this);

        btnSubmitCha = FindViewById<Button>(Resource.Id.btnCreateCha);
```

```
    btnSubmitCha.SetOnClickListener(this);

    inv1 = FindViewById<Button>(Resource.Id.btnInvite1);
    inv1.SetOnClickListener(this);

    inv2 = FindViewById<Button>(Resource.Id.btnInvite2);
    inv2.SetOnClickListener(this);


    etTeamName = FindViewById<EditText>(Resource.Id.etNewTeamName);


    if (teambuilt == null)
    {
        teambuilt = new String[5];
    }



    if (challengesChosenChalYear != null)
    {

        setSelSpinner(challengesChosenChalYear);

        spinner.Enabled = false;

        inv1.Enabled = false;
        inv2.Enabled = false;


    }




}

public void setSelSpinner(string y)
{
    if (y.Equals("2019"))
    {
        spinner.SetSelection(0);
    }
    else if (y.Equals("2018"))
    {
        spinner.SetSelection(1);
    }
    else if (y.Equals("2017"))
    {
        spinner.SetSelection(2);
    }
    else if (y.Equals("2016"))
    {
        spinner.SetSelection(3);
    }
    else if (y.Equals("2015"))
    {
        spinner.SetSelection(4);
    }
    else if (y.Equals("2014"))
```

```
            {
                spinner.SetSelection(5);
            }
            else if (y.Equals("2013"))
            {
                spinner.SetSelection(6);
            }
        }

        private void spinner_ItemSelected(object sender, AdapterView.ItemSelectedEventArgs e)
        {
            Spinner spinner = (Spinner)sender;
            valYear = string.Format("{0}", spinner.GetItemAtPosition(e.Position));
            Toast.MakeText(this, valYear, ToastLength.Long).Show();

        }


        public static bool isGoodCha()
        {
            for(int i =0; i<teambuilt.Length; i++)
            {
                if(teambuilt[i] == null || teambuilt[i].Equals(""))
                {
                    return false;
                }
            }
            if (etTeamName.Text == null || etTeamName.Text.Equals(""))
            {
                return false;
            }

            if(challengesChosenChalYear == null)
            {
                if (usr1 == null || usr2 == null)
                {
                    return false;
                }

                if (usr1.Equals(usr2))
                {
                    return false;
                }

            }

            //check isExists user


            return true;

        }


    }
}
```

chalAdapter.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.Graphics;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using My_Fantasy_Mychallenge;

namespace My_Fantasy
{
    class ChalAdapter : BaseAdapter<Mychallenge>
    {
        Android.Content.Context context;

        List<Mychallenge> objects;

        //Android.Views.LayoutInflater layoutInflater;

        public ChalAdapter(Android.Content.Context context,
System.Collections.Generic.List<Mychallenge> objects)

        {



            this.context = context;



            this.objects = objects;


            //layoutInflater = ((Challenges)context).LayoutInflater;
            //Android.Views.View view =
layoutInflater.Inflate(Resource.Layout.oneItemChallenge, parent, false);

        }



        public void setList(System.Collections.Generic.List<Mychallenge> objects)
        {
            this.objects = objects;
        }



        public override long GetItemId(int position)
        {
            return position;
        }


        public override int Count
```

```
        {
            get { return this.objects.Count; }
        }


        public override Mychallenge this[int position]
        {
            get { return this.objects[position]; }
        }


        public override View GetView(int position, View convertView, ViewGroup parent)
        {
            Android.Views.LayoutInflater layoutInflater =
((Challenges)context).LayoutInflater;
            Android.Views.View view =
layoutInflater.Inflate(Resource.Layout.oneItemChallenge, parent, false);
            TextView tvUser = view.FindViewById<TextView>(Resource.Id.tvUser);
            TextView tvYear =
view.FindViewById<TextView>(Resource.Id.tvTypeChallenge);
            TextView tvScoreCreator =
view.FindViewById<TextView>(Resource.Id.tvScoreCreator);
            TextView tvPer1 = view.FindViewById<TextView>(Resource.Id.tvPer1);
            TextView tvScorePer1 =
view.FindViewById<TextView>(Resource.Id.tvScorePer1);
            TextView tvPer2 = view.FindViewById<TextView>(Resource.Id.tvPer2);
            TextView tvScorePer2 =
view.FindViewById<TextView>(Resource.Id.tvScorePer2);
            Mychallenge temp = objects[position];

            if (temp != null)
            {
                tvUser.Text = temp.creator.user;

                if (temp.creator.user.Equals(MainActivity.usr))
                {
                    tvUser.SetTextColor(Color.Blue);
                }

                tvYear.Text =  temp.year + "|";
                tvScoreCreator.Text = "|" + temp.creator.points;
                tvPer1.Text = "|" + temp.Team1.user;
                tvScorePer1.Text = "|" + temp.Team1.points;
                tvPer2.Text = "|" + temp.Team2.user;
                tvScorePer2.Text = "|" + temp.Team2.points;



                if(temp.Team1.points!= 0 && temp.Team2.points != 0)
                {
                    int maxPts = Math.Max(Math.Max(temp.Team1.points,
temp.Team2.points), temp.creator.points);

                    if(temp.creator.points == maxPts)
                    {
                        tvScoreCreator.SetTextColor(Color.Green);
                        //tvUser.SetTextColor(Color.Green);
                    }
                    if (temp.Team1.points == maxPts)
                    {
                        tvPer1.SetTextColor(Color.Green);
                        tvScorePer1.SetTextColor(Color.Green);
```

```
                    }
                    if (temp.Team2.points == maxPts)
                    {
                        tvPer2.SetTextColor(Color.Green);
                        tvScorePer2.SetTextColor(Color.Green);
                    }


                }

            }

            return view;


        }

    }
}
```

Challenges.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.Media;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.Support.V4.App;
using TaskStackBuilder = Android.Support.V4.App.TaskStackBuilder;
using Newtonsoft.Json;
using My_Fantasy_Mychallenge;
using System.Timers;
//using Java.Lang;

namespace My_Fantasy
{
    [Activity(Label = "Challenges")]
    public class Challenges : Activity, Android.Views.View.IOnClickListener,
ListView.IOnItemClickListener
    {

        Button newCha;


        Button myRefresh;

        static ListView requests;
        //Android.Views.LayoutInflater locallayoutInflater;
        int count = 0;
        MediaPlayer p;
        static List<Mychallenge> l;
        static ChalAdapter chalAdapter;
        public static bool isNewChal;
```

```
        public static string chosenChalYear = "";

        static readonly int NOTIFICATION_ID = 1000;
        static readonly string CHANNEL_ID = "location_notification";
        internal static readonly string COUNT_KEY = "count";

        Handler mHandler;

        public Challenges()
        {

        }

        public void OnClick(View v)
        {

            if (v == newCha)
            {
                aTimer.Stop();
                Intent intent = new Intent(this, typeof(Building));
                isNewChal = true;
                StartActivityForResult(intent, 3);


                //StartActivity(intent);
            }

            if (v == myRefresh)
            {

                updateTable();
            }



        }


        protected override void OnActivityResult(int requestCode, [GeneratedEnum]
Result resultCode, Intent data)
        {
            base.OnActivityResult(requestCode, resultCode, data);

            RunOnUiThread(() =>
            {
                myRefresh.CallOnClick();
            });


            aTimer.Start();
        }

        private void updateTable()
        {
            ConnectToSer c = new ConnectToSer();
            string ans = c.CallMyServer("challenges", MainActivity.usr);

            List<Mychallenge> deserializedProduct =
JsonConvert.DeserializeObject<List<Mychallenge>>(ans);

            l.Clear();
```

```
            l.AddRange(deserializedProduct);

            //chalAdapter
            //requests.Adapter
            //chalAdapter = new ChalAdapter(source, l);
            chalAdapter.setList(l);
            requests.Adapter = chalAdapter;
        }


        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);

            // Create your application here

            SetContentView(Resource.Layout.challengesNew);

            newCha = FindViewById<Button>(Resource.Id.btnNewChallenge);
            myRefresh = FindViewById<Button>(Resource.Id.btnRefresh);
            myRefresh.SetOnClickListener(this);
            myRefresh.Visibility = ViewStates.Invisible;
            //myCha = FindViewById<Button>(Resource.Id.btnYourCahllenge);
            requests = FindViewById<ListView>(Resource.Id.lvChallenges);

            //myCha.SetOnClickListener(this);
            newCha.SetOnClickListener(this);

            Intent intent = new Intent(ApplicationContext,
typeof(MediaPlayerService));
            intent.SetAction(MediaPlayerService.ACTION_PLAY);
            StartService(intent);


            createChannel();


            ConnectToSer c = new ConnectToSer();
            string ans = "";
            try
            {
                ans = c.CallMyServer("challenges", MainActivity.usr);

                List<Mychallenge> deserializedProduct =
JsonConvert.DeserializeObject<List<Mychallenge>>(ans);

                requests = FindViewById<ListView>(Resource.Id.lvChallenges);

                //LVplayers.check = false;

                l = new List<Mychallenge>();
                l.AddRange(deserializedProduct);

                chalAdapter = new ChalAdapter(this, l);
                requests.Adapter = chalAdapter;

                requests.OnItemClickListener = this;
                //get is playing by intent
                //enabled false buttons and lvChallenges if is playing in challense
                //you can only see the status of the current challenge and the other
players
```

```
                SetTimer();
            }
            catch
            {
                AlertDialog.Builder builder = new AlertDialog.Builder(this);

                builder.SetTitle("Error connceting to server");

                builder.SetMessage("Try again later");

                builder.SetCancelable(true);

                //builder.SetPositiveButton("OK", OkAction);

                //builder.SetNegativeButton("cancel", CancelAction);

                AlertDialog dialog = builder.Create();

                dialog.Show();
            }


        }



        public override void OnBackPressed()
        {
            // disables you come back to register window
            return;


        }

        public void OnItemClick(AdapterView parent, View view, int position, long id)
        {



            //throw new System.NotImplementedException();

            Intent intent = new Intent(this, typeof(Building));

            Mychallenge temp = Challenges.l[position];
            chosenChalYear = temp.year;
            isNewChal = false;

            if (temp.creator.user.Equals(MainActivity.usr))
            {
                Toast.MakeText(this, "cannot pick yourself", ToastLength.Long).Show();
                return;

            }

            if (temp.creator.points != 0 && temp.Team1.points != 0 &&
temp.Team2.points != 0)
            {
                Toast.MakeText(this, "This challenge has already been done",
ToastLength.Long).Show();
                return;
            }
```

```
            //intent.PutExtra("pos", position);

            //intent.PutExtra("isNewChal", isNewChal);
            intent.PutExtra("chosenChalYear", chosenChalYear);
            intent.PutExtra("challengeID", temp.challengeID);
            StartActivityForResult(intent, 1);


            //StartActivity(intent);

        }




        private void createChannel()
        {
            if (Build.VERSION.SdkInt < BuildVersionCodes.O)
            {
                // Notification channels are new in API 26 (and not a part of the
                // support library). There is no need to create a notification
                // channel on older versions of Android.
                return;
            }

            var name = Resources.GetString(Resource.String.channel_name);
            var description = GetString(Resource.String.channel_description);
            var channel = new NotificationChannel(CHANNEL_ID, name,
NotificationImportance.Default)
            {
                Description = description
            };

            var notificationManager =
(NotificationManager)GetSystemService(NotificationService);
            notificationManager.CreateNotificationChannel(channel);
        }


        private static System.Timers.Timer aTimer;

        private void SetTimer()
        {
            // Create a timer with a two second interval.
            aTimer = new System.Timers.Timer(3000);
            // Hook up the Elapsed event for the timer.
            aTimer.Elapsed += OnTimedEvent;
            aTimer.AutoReset = false;
            aTimer.Enabled = true;



        }

        void MyElapsedMethod(object sender, ElapsedEventArgs e,
Android.Content.Context myThis)
        {

            RunOnUiThread(() =>
            {
                myRefresh.CallOnClick();
```

```
            });

            //Console.WriteLine("The Elapsed event was raised at {0:HH:mm:ss.fff}",
                              //   e.SignalTime);


            aTimer.Start();

        }

        private void OnTimedEvent(Object source, ElapsedEventArgs e)
        {
            RunOnUiThread(() =>
            {
                myRefresh.CallOnClick();
            });

            Console.WriteLine("The Elapsed event was raised at {0:HH:mm:ss.fff}",
                              e.SignalTime);


            aTimer.Start();
        }


    }
}
```

chalsOBJS.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

namespace My_Fantasy_Mychallenge
{


    public class Rootobject
    {
        public Mychallenge[] Property1 { get; set; }
    }

    public class Mychallenge
    {
        public Team1 Team1 { get; set; }
        public Team2 Team2 { get; set; }
        public string challengeID { get; set; }
        public Creator creator { get; set; }
        public int status { get; set; }
        public string year { get; set; }
    }
```

```
    public class Team1
    {
        public string name { get; set; }
        public object ownerId { get; set; }
        public string[] players { get; set; }
        public int points { get; set; }
        public string teamId { get; set; }
        public string user { get; set; }
        public string year { get; set; }
    }

    public class Team2
    {
        public string name { get; set; }
        public object ownerId { get; set; }
        public string[] players { get; set; }
        public int points { get; set; }
        public string teamId { get; set; }
        public string user { get; set; }
        public string year { get; set; }
    }

    public class Creator
    {
        public string name { get; set; }
        public object ownerId { get; set; }
        public string[] players { get; set; }
        public int points { get; set; }
        public string teamId { get; set; }
        public string user { get; set; }
        public string year { get; set; }
    }


}
```

ConnectToSer.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;



namespace My_Fantasy
{
    class ConnectToSer
    {

        public static string myurl = "http://192.168.3.106:5000/";
```

```
        public string CallMyServer(string function, string param)
        {

            string xurl = "";
            if (string.IsNullOrEmpty(param))
            {
                xurl = myurl + function;
            }
            else
            {
                xurl = myurl + function + "/" + param;
            }
            //Toast.MakeText(this, "b4 call url = " + xurl, ToastLength.Long).Show();

            var request = (HttpWebRequest)WebRequest.Create(xurl);

            request.Method = "GET";
            //request.UserAgent = RequestConstants.UserAgentValue;
            request.AutomaticDecompression = DecompressionMethods.Deflate |
DecompressionMethods.GZip;


            var content = string.Empty;
            request.Timeout = 10000;
            using (var response = (HttpWebResponse)request.GetResponse())
            {
                using (var stream = response.GetResponseStream())
                {
                    using (var sr = new StreamReader(stream))
                    {
                        content = sr.ReadToEnd();

                    }
                }
            }



            return content;
        }


    }
}
```

lvPlayers.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Newtonsoft.Json;
```

```
namespace My_Fantasy
{
    [Activity(Label = "LVplayers")]
    public class LVplayers : Activity, ListView.IOnItemClickListener
    {
        ListView lv;
        static List<Standard> Allpls;
        static List<Standard> AllplsByPos;
        static bool check = false;
        static string chosenYearInNew;

        string myAnswer = "";

        static string posChosen = "";
        string valYear = "2019";
        String[] teambuilt;


        protected override void OnCreate(Bundle savedInstanceState)
        {
            posChosen = Intent.GetStringExtra("posChosen");
            valYear = Intent.GetStringExtra("valYear");
            teambuilt = Intent.GetStringArrayExtra("teambuilt");



            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.listPlayers);
            lv = FindViewById<ListView>(Resource.Id.lv);
            ConnectToSer c = new ConnectToSer();
            string ans = c.CallMyServer("players", valYear);
            RootobjectPlayers deserializedProduct =
JsonConvert.DeserializeObject<RootobjectPlayers>(ans);

            lv = FindViewById<ListView>(Resource.Id.lv);

            Allpls = new System.Collections.Generic.List<Standard>();


            foreach (Standard item in deserializedProduct.league.standard)
            {
                if ( item.teams.Length != 0)
                {
                    Allpls.Add(item);
                }

            }


            //Allpls.AddRange(deserializedProduct.league.standard);
            AllplsByPos = new System.Collections.Generic.List<Standard>();
            AllplsByPos.AddRange(getByPos(Allpls));

            PlayerAdapter p = new PlayerAdapter(this, AllplsByPos);

            lv.Adapter = p;


            lv.OnItemClickListener = this;

        }
```

```
        public static List<Standard> getByPos(List<Standard> l)
        {
            List<Standard> newByPos = new List<Standard>();
            char pos = posChosen[posChosen.Length-1];

            foreach (Standard i in l)
            {
                if (i.pos.Contains(pos))
                {
                    newByPos.Add(i);
                }
            }
            return newByPos;
        }

        public void OnItemClick(AdapterView parent, View view, int position, long id)
        {
            Standard temp = LVplayers.AllplsByPos[position];

            if (!teambuilt.Contains(temp.personId))
            {
                if (posChosen.Equals("PG"))
                {
                    teambuilt[0] = temp.personId;

                }else if (posChosen.Equals("SG"))
                {
                    teambuilt[1] = temp.personId;
                }
                else if (posChosen.Equals("SF"))
                {
                    teambuilt[2] = temp.personId;
                }
                else if (posChosen.Equals("PF"))
                {
                    teambuilt[3] = temp.personId;

                }
                else
                {
                    teambuilt[4] = temp.personId;
                }

                check = true;
                chosenYearInNew = valYear;
                //Android.Content.Intent intent = new Android.Content.Intent(this,
typeof(Building));

                //StartActivityForResult(intent, 1);
                Intent intent = new Intent();
                intent.PutExtra("playerId", temp.personId);
                intent.PutExtra("playerFullName", temp.firstName + " " +
temp.lastName);
                intent.PutExtra("posChosen", posChosen);

                SetResult(Result.Ok, intent);

                Finish();
            }
            else
            {
```

```
                Toast.MakeText(this, "You have already chosen that player!",
ToastLength.Long).Show();
            }


        }

    }
}
```

LVUsers.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Newtonsoft.Json;

using My_Fantasy_UsersOBJ;


namespace My_Fantasy
{
    [Activity(Label = "LVUsers")]
    public class LVUsers : Activity, ListView.IOnItemClickListener
    {
        ListView lv;
        List<SingleUser> allUsers;



        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.listUsers);
            // Create your application here

            lv = FindViewById<ListView>(Resource.Id.lv);
            ConnectToSer c = new ConnectToSer();
            string ans = c.CallMyServer("getUsers", "");
            //List<My_Fantasy_UsersOBJ.Rootobject> deserializedProduct =
JsonConvert.DeserializeObject<My_Fantasy_UsersOBJ.Rootobject>(ans);

            allUsers = JsonConvert.DeserializeObject<List<SingleUser>>(ans);

            removeMe();

            lv = FindViewById<ListView>(Resource.Id.lvUsers);

            //allUsers = new System.Collections.Generic.List<User>();

            //allUsers.AddRange(deserializedProduct.users);
```

```
            UserAdapter p = new UserAdapter(this, allUsers);

            lv.Adapter = p;

            lv.OnItemClickListener = this;


        }

        public void OnItemClick(AdapterView parent, View view, int position, long id)
        {

            SingleUser temp = allUsers[position];
            Intent intent = new Intent();
            intent.PutExtra("name", temp.name);
            intent.PutExtra("userId", temp.userId);

            SetResult(Result.Ok, intent);

            Finish();

        }

        public void removeMe()
        {
            //remove the user. He cannot pick himself
            List<SingleUser> newL = new List<SingleUser>();
            foreach (SingleUser item in allUsers)
            {
                if (!item.name.Equals(MainActivity.usr))
                {
                    newL.Add(item);
                }
            }
            allUsers = newL;
        }

    }
}
```

MainActivity.cs

```
using Android.App;
using Android.OS;
using Android.Widget;
using Android.Views;
using System.IO;
using System.Text;
using Android.Content;

namespace My_Fantasy
{
    [Activity(Label = "@string/app_name", Theme = "@style/AppTheme")]
    public class MainActivity : Activity, Android.Views.View.IOnClickListener
    {

        EditText newUserName;
        Button submitNewUser;
        public string str;
        int count = 0;
        public static string usr;
```

```
        public void OnClick(View v)
        {
            if (v == submitNewUser)
            {

                string str = newUserName.Text;
                if (!str.Equals("") )
                {
                    ConnectToSer c = new ConnectToSer();

                    string ans = c.CallMyServer("user", str);

                    if (ans.Equals("Added"))
                    {
                        try
                        {
                            using (Stream stream = OpenFileOutput("username.xml",
FileCreationMode.Private))
                            {

                                if (str != null)
                                {

                                    try
                                    {

                                        stream.Write(Encoding.UTF8.GetBytes(str), 0,
str.Length);


                                        stream.Close();

                                        Toast.MakeText(this, "saved!",
ToastLength.Long).Show();
                                        MainActivity.usr = str;
                                        Android.Content.Intent i = new
Android.Content.Intent(this, typeof(Challenges));
                                        StartActivityForResult(i, 1);

                                    }

                                    catch (Java.IO.IOException e)
                                    {

                                        e.PrintStackTrace();

                                    }
                                }

                                Android.Content.Intent intent = new
Android.Content.Intent(this, typeof(Challenges));

                                StartActivityForResult(intent, 1);//1 means go to test


                            }
                        }
                        catch (Java.IO.FileNotFoundException e)
                        {
                            e.PrintStackTrace();
                        }
```

```
                    }
                    else
                    {
                        Toast.MakeText(this, "already exists!",
ToastLength.Long).Show();
                        newUserName.Text = "";
                    }

                }

            }

        }

        protected override void OnCreate(Bundle savedInstanceState)
        {



            base.OnCreate(savedInstanceState);
            // Set our view from the "main" layout resource
            SetContentView(Resource.Layout.Harshama);

            bool b = IsReg();

            if (b)
            {

                Android.Content.Intent intent = new Android.Content.Intent(this,
typeof(Challenges));
                StartActivityForResult(intent, 1);
            }


            newUserName = FindViewById<EditText>(Resource.Id.etNewUname);
            submitNewUser = FindViewById<Button>(Resource.Id.btnNewUser);

            submitNewUser.SetOnClickListener(this);


        }

        public override void OnBackPressed()
        {
            // disables you come back
            return;

        }

        public bool IsReg()
        {

            try

            {

                //using (var i = new
StreamReader(OpenFileInput(NumeroCartaoFilename)))

                using (Stream inTo = OpenFileInput("username.xml"))
```

```
                try

                {

                    byte[] buffer = new byte[4096];

                    int num  = inTo.Read(buffer, 0, buffer.Length);

                    str = System.Text.Encoding.UTF8.GetString(buffer);

                    str = str.Substring(0, num);

                    inTo.Close();

                    if (str != null) {
                        usr = str;
                        return true;
                    }

                    return false;
                }

                catch (Java.IO.IOException e)

                {

                    e.PrintStackTrace();
                    return false;

                }


            }

            catch (Java.IO.FileNotFoundException e)

            {

                e.PrintStackTrace();
                return false;

            }



        }



    }
}
```

NBAPlayerOBJ.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

namespace My_Fantasy
{

    public class RootobjectPlayers
    {
        public _Internal _internal { get; set; }
        public League league { get; set; }
    }

    public class _Internal
    {
        public string pubDateTime { get; set; }
        public string xslt { get; set; }
        public string eventName { get; set; }
    }

    public class League
    {
        public Standard[] standard { get; set; }
        public Africa[] africa { get; set; }
        public Sacramento[] sacramento { get; set; }
        public Vega[] vegas { get; set; }
        public Utah[] utah { get; set; }
    }

    public class Standard
    {
        public string firstName { get; set; }
        public string lastName { get; set; }
        public string personId { get; set; }
        public string teamId { get; set; }
        public string jersey { get; set; }
        public bool isActive { get; set; }
        public string pos { get; set; }
        public string heightFeet { get; set; }
        public string heightInches { get; set; }
        public string heightMeters { get; set; }
        public string weightPounds { get; set; }
        public string weightKilograms { get; set; }
        public string dateOfBirthUTC { get; set; }
        public Team[] teams { get; set; }
        public Draft draft { get; set; }
        public string nbaDebutYear { get; set; }
        public string yearsPro { get; set; }
        public string collegeName { get; set; }
        public string lastAffiliation { get; set; }
        public string country { get; set; }
    }
```

```
public class Draft
{
    public string teamId { get; set; }
    public string pickNum { get; set; }
    public string roundNum { get; set; }
    public string seasonYear { get; set; }
}

public class Team
{
    public string teamId { get; set; }
    public string seasonStart { get; set; }
    public string seasonEnd { get; set; }
}

public class Africa
{
    public string firstName { get; set; }
    public string lastName { get; set; }
    public string personId { get; set; }
    public string teamId { get; set; }
    public string jersey { get; set; }
    public bool isActive { get; set; }
    public string pos { get; set; }
    public string heightFeet { get; set; }
    public string heightInches { get; set; }
    public string heightMeters { get; set; }
    public string weightPounds { get; set; }
    public string weightKilograms { get; set; }
    public string dateOfBirthUTC { get; set; }
    public Team1[] teams { get; set; }
    public Draft1 draft { get; set; }
    public string nbaDebutYear { get; set; }
    public string yearsPro { get; set; }
    public string collegeName { get; set; }
    public string lastAffiliation { get; set; }
    public string country { get; set; }
}

public class Draft1
{
    public string teamId { get; set; }
    public string pickNum { get; set; }
    public string roundNum { get; set; }
    public string seasonYear { get; set; }
}

public class Team1
{
    public string teamId { get; set; }
    public string seasonStart { get; set; }
    public string seasonEnd { get; set; }
}

public class Sacramento
{
    public string firstName { get; set; }
    public string lastName { get; set; }
    public string personId { get; set; }
    public string teamId { get; set; }
    public string jersey { get; set; }
```

```
    public bool isActive { get; set; }
    public string pos { get; set; }
    public string heightFeet { get; set; }
    public string heightInches { get; set; }
    public string heightMeters { get; set; }
    public string weightPounds { get; set; }
    public string weightKilograms { get; set; }
    public string dateOfBirthUTC { get; set; }
    public Team2[] teams { get; set; }
    public Draft2 draft { get; set; }
    public string nbaDebutYear { get; set; }
    public string yearsPro { get; set; }
    public string collegeName { get; set; }
    public string lastAffiliation { get; set; }
    public string country { get; set; }
}

public class Draft2
{
    public string teamId { get; set; }
    public string pickNum { get; set; }
    public string roundNum { get; set; }
    public string seasonYear { get; set; }
}

public class Team2
{
    public string teamId { get; set; }
    public string seasonStart { get; set; }
    public string seasonEnd { get; set; }
}

public class Vega
{
    public string firstName { get; set; }
    public string lastName { get; set; }
    public string personId { get; set; }
    public string teamId { get; set; }
    public string jersey { get; set; }
    public bool isActive { get; set; }
    public string pos { get; set; }
    public string heightFeet { get; set; }
    public string heightInches { get; set; }
    public string heightMeters { get; set; }
    public string weightPounds { get; set; }
    public string weightKilograms { get; set; }
    public string dateOfBirthUTC { get; set; }
    public Team3[] teams { get; set; }
    public Draft3 draft { get; set; }
    public string nbaDebutYear { get; set; }
    public string yearsPro { get; set; }
    public string collegeName { get; set; }
    public string lastAffiliation { get; set; }
    public string country { get; set; }
}

public class Draft3
{
    public string teamId { get; set; }
    public string pickNum { get; set; }
    public string roundNum { get; set; }
    public string seasonYear { get; set; }
```

```
    }

    public class Team3
    {
        public string teamId { get; set; }
        public string seasonStart { get; set; }
        public string seasonEnd { get; set; }
    }

    public class Utah
    {
        public string firstName { get; set; }
        public string lastName { get; set; }
        public string personId { get; set; }
        public string teamId { get; set; }
        public string jersey { get; set; }
        public bool isActive { get; set; }
        public string pos { get; set; }
        public string heightFeet { get; set; }
        public string heightInches { get; set; }
        public string heightMeters { get; set; }
        public string weightPounds { get; set; }
        public string weightKilograms { get; set; }
        public string dateOfBirthUTC { get; set; }
        public object[] teams { get; set; }
        public Draft4 draft { get; set; }
        public string nbaDebutYear { get; set; }
        public string yearsPro { get; set; }
        public string collegeName { get; set; }
        public string lastAffiliation { get; set; }
        public string country { get; set; }
    }

    public class Draft4
    {
        public string teamId { get; set; }
        public string pickNum { get; set; }
        public string roundNum { get; set; }
        public string seasonYear { get; set; }
    }

}
```

Patiah.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Timers;
using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

namespace My_Fantasy
{
    [Activity(Label = "MyFantasy", MainLauncher = true)]
```

```
    public class patiah : Activity
    {

        private static System.Timers.Timer aTimer;

        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.patiah);
            // Create your application here

            aTimer = new System.Timers.Timer(5000);
            // Hook up the Elapsed event for the timer.
            aTimer.Elapsed += OnTimedEvent;
            aTimer.AutoReset = false;
            aTimer.Enabled = true;

            aTimer.Start();


        }


        private void OnTimedEvent(Object source, ElapsedEventArgs e)
        {


            Console.WriteLine("The Elapsed event was raised at {0:HH:mm:ss.fff}",
                              e.SignalTime);

            Android.Content.Intent intent = new Android.Content.Intent(this,
typeof(MainActivity));
            StartActivity(intent);

        }

    }
}
```

PlayerAdapter.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

namespace My_Fantasy
{
    class PlayerAdapter : BaseAdapter<Standard>
    {
        Android.Content.Context context;
```

```
        List<Standard> objects;


        public PlayerAdapter(Android.Content.Context context,
System.Collections.Generic.List<Standard> objects)

        {



            this.context = context;



            this.objects = objects;



        }




        public List<Standard> GetList()
        {
            List<Standard> fs = new List<Standard>();

            foreach (var i in this.objects)
            {
                if (i.pos == "F")
                {
                    fs.Add(i);
                }
            }


            return fs;
        }


        public override long GetItemId(int position)
        {
            return position;
        }


        public override int Count
        {
            get { return this.objects.Count; }
        }

        public override Standard this[int position]
        {
            get { return this.objects[position]; }
        }


        public override View GetView(int position, View convertView, ViewGroup parent)
        {
```

```
            Android.Views.LayoutInflater layoutInflater =
((LVplayers)context).LayoutInflater;
            Android.Views.View view = layoutInflater.Inflate(Resource.Layout.oneItemP,
parent, false);
            TextView tvTitle = view.FindViewById<TextView>(Resource.Id.tvFull);

            Standard temp = objects[position];

            if (temp != null)
            {
                tvTitle.Text = temp.firstName + " " + temp.lastName;
            }

            return view;



        }
    }
}
```

StreamingBackgroundService.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.Media;
using Android.Media.Session;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

namespace My_Fantasy
{
    [Service]
    public class MediaPlayerService : IntentService
    {
        public static String ACTION_PLAY = "action_play";
        public static String ACTION_PAUSE = "action_pause";
        public static String ACTION_REWIND = "action_rewind";
        public static String ACTION_FAST_FORWARD = "action_fast_foward";
        public static String ACTION_NEXT = "action_next";
        public static String ACTION_PREVIOUS = "action_previous";
        public static String ACTION_STOP = "action_stop";

        private static MediaPlayer mMediaPlayer;
        private static MediaSessionManager mManager;
        private static MediaSession mSession;
        private static Android.Media.Session.MediaController mController;


        private static String CHANNEL_ID = "MediaPlayerService";

        public MediaPlayerService() : base("MediaPlayerService")
        {
            //InitMediaSession();
```

```
        }

        [return: GeneratedEnum]
        public override StartCommandResult OnStartCommand(Intent intent,
[GeneratedEnum] StartCommandFlags flags, int startId)
        {
            InitMediaSession();
            return base.OnStartCommand(intent, flags, startId);
        }

        private void InitMediaSession()
        {

            if (mMediaPlayer == null)
            {
                mMediaPlayer = new MediaPlayer();
                mMediaPlayer.SetAudioStreamType(Stream.Music);
                mMediaPlayer.Prepared += (sender, args) => mMediaPlayer.Start();
                mMediaPlayer.Completion += (sender, args) => mMediaPlayer.Pause();
                SetPlayer(Resource.Raw.Halleluka);

                mMediaPlayer.Looping = true;

                mManager = (MediaSessionManager)GetSystemService(MediaSessionService);
                mSession = new MediaSession(this, "sample session");
                mController = mSession.Controller;

                mSession.SetCallback(new CustomCallback(this));
            }
        }

        private void SetPlayer(int mp3)
        {
            String uri = "android.resource://My_Fantasy.My_Fantasy/Raw/" + mp3;
            mMediaPlayer.SetDataSource(ApplicationContext,
Android.Net.Uri.Parse(uri));
            mMediaPlayer.Prepare();
        }

        class CustomCallback : MediaSession.Callback
        {

            MediaPlayerService service;
            public CustomCallback(MediaPlayerService service) : base()
            {
                this.service = service;
            }
            public override void OnFastForward()
            {
                base.OnFastForward();
            }

            public override void OnPause()
            {
                //base.OnPause();

                if (mMediaPlayer != null && mMediaPlayer.IsPlaying)
                {
                    mMediaPlayer.Pause();
                }
```

```
service.buildNotification(service.generateAction(Android.Resource.Drawable.IcMediaPlay
, "Play", ACTION_PLAY));
            }

            public override void OnPlay()
            {
                //base.OnPlay();
                if (mMediaPlayer != null && !mMediaPlayer.IsPlaying)
                {
                    mMediaPlayer.Start();
                }



service.buildNotification(service.generateAction(Android.Resource.Drawable.IcMediaPaus
e, "Pause", ACTION_PAUSE));
            }

            public override void OnRewind()
            {
                base.OnRewind();
            }

            public override void OnSeekTo(long pos)
            {
                base.OnSeekTo(pos);
            }

            public override void OnSetRating(Rating rating)
            {
                base.OnSetRating(rating);
            }

            public override void OnSkipToNext()
            {
                base.OnSkipToNext();

service.buildNotification(service.generateAction(Android.Resource.Drawable.IcMediaPaus
e, "Pause", ACTION_PAUSE));
            }

            public override void OnSkipToPrevious()
            {
                base.OnSkipToPrevious();

service.buildNotification(service.generateAction(Android.Resource.Drawable.IcMediaPaus
e, "Pause", ACTION_PAUSE));
            }

            public override void OnStop()
            {
                base.OnStop();

                NotificationManager notificationManager =
NotificationManager.FromContext(this.service);
                notificationManager.Cancel(1);
                Intent intent = new Intent(service.ApplicationContext,
typeof(MediaPlayerService));
                service.StopService(intent);
            }
        }
```

```
        protected override void OnHandleIntent(Intent intent)
        {
            if (intent == null || intent.Action == null)
                return;

            String action = intent.Action;

            if (action.Equals(ACTION_PLAY))
            {
                mController.GetTransportControls().Play();
                //mMediaPlayer.Start();
            }
            else if (action.Equals(ACTION_PAUSE))
            {
                mController.GetTransportControls().Pause();
                //mMediaPlayer.Pause();
            }
            else if (action.Equals(ACTION_FAST_FORWARD))
            {
                mController.GetTransportControls().FastForward();
            }
            else if (action.Equals(ACTION_REWIND))
            {
                mController.GetTransportControls().Rewind();
            }
            else if (action.Equals(ACTION_PREVIOUS))
            {
                mController.GetTransportControls().SkipToPrevious();
            }
            else if (action.Equals(ACTION_NEXT))
            {
                mController.GetTransportControls().SkipToNext();
            }
            else if (action.Equals(ACTION_STOP))
            {
                mController.GetTransportControls().Stop();
                mMediaPlayer.Stop();
            }
        }

        private void buildNotification(Notification.Action action)
        {
            Notification.MediaStyle style = new Notification.MediaStyle();

            Intent intent = new Intent(ApplicationContext,
typeof(MediaPlayerService));
            intent.SetAction(ACTION_STOP);
            PendingIntent pendingIntent = PendingIntent.GetService(ApplicationContext,
1, intent, 0);

            createChannel();
            Notification.Builder builder = new Notification.Builder(this, CHANNEL_ID)
                .SetSmallIcon(Resource.Drawable.a1)
                .SetContentTitle("Media Title")
                .SetContentText("Media Artist")
                .SetDeleteIntent(pendingIntent)
                .SetStyle(style)
                .SetOnlyAlertOnce(true);
```

```
builder.AddAction(generateAction(Android.Resource.Drawable.IcMediaPrevious,
"Previous", ACTION_PREVIOUS));
            builder.AddAction(generateAction(Android.Resource.Drawable.IcMediaRew,
"Rewind", ACTION_REWIND));
            builder.AddAction(action);
            builder.AddAction(generateAction(Android.Resource.Drawable.IcMediaFf,
"Fast Foward", ACTION_FAST_FORWARD));
            builder.AddAction(generateAction(Android.Resource.Drawable.IcMediaNext,
"Next", ACTION_NEXT));

            NotificationManager notificationManager =
NotificationManager.FromContext(this);
            notificationManager.Notify(1, builder.Build());

        }

        private Notification.Action generateAction(int icon, String title, String
intentAction)
        {
            Intent intent = new Intent(ApplicationContext,
typeof(MediaPlayerService));
            intent.SetAction(intentAction);
            PendingIntent pendingIntent = PendingIntent.GetService(ApplicationContext,
1, intent, 0);
            return new Notification.Action.Builder(icon, title,
pendingIntent).Build();
        }

        public override bool OnUnbind(Intent intent)
        {
            mSession.Release();
            return base.OnUnbind(intent);
        }

        private void createChannel()
        {
            if (Build.VERSION.SdkInt < BuildVersionCodes.O)
            {
                // Notification channels are new in API 26 (and not a part of the
                // support library). There is no need to create a notification
                // channel on older versions of Android.
                return;
            }

            var name = Resources.GetString(Resource.String.channel_name);
            var description = GetString(Resource.String.channel_description);
            var channel = new NotificationChannel(CHANNEL_ID, name,
NotificationImportance.Default)
            {
                Description = description
            };

            var notificationManager =
(NotificationManager)GetSystemService(NotificationService);
            notificationManager.CreateNotificationChannel(channel);
        }
    }
}
```

UserAdapter.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using My_Fantasy_UsersOBJ;

namespace My_Fantasy
{
    class UserAdapter : BaseAdapter<SingleUser>
    {
        Android.Content.Context context;

        List<SingleUser> objects;



        public UserAdapter(Android.Content.Context context,
System.Collections.Generic.List<SingleUser> objects)

        {
            this.context = context;

            this.objects = objects;

        }

        public override long GetItemId(int position)
        {
            return position;
        }


        public override int Count
        {
            get { return this.objects.Count; }
        }



        public override SingleUser this[int position]
        {
            get { return this.objects[position]; }
        }


        public override View GetView(int position, View convertView, ViewGroup parent)
        {
            Android.Views.LayoutInflater layoutInflater =
((LVUsers)context).LayoutInflater;
            Android.Views.View view = layoutInflater.Inflate(Resource.Layout.oneUser,
parent, false);
            TextView tvUser = view.FindViewById<TextView>(Resource.Id.tvUsername);
```

```
            SingleUser temp = objects[position];

            if (temp != null)
            {
                tvUser.Text = temp.name;

            }

            return view;


        }

    }
}
```

UserOBJ.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

namespace My_Fantasy_UsersOBJ
{


    public class SingleUser
    {
        public string name { get; set; }
        public string userId { get; set; }
    }

}
```