נושא הפרויקט:

משחק THE IMPOSTOR



עבודת גמר תכנון ותכנות מערכות 883599

חלופה: תכנות טלפונים חכמים

מגיש/ה: דנה עיון

ת.ז: 214779019

מנחה: רועי אמיגה

תיכון אחד העם

פתח תקווה, תשפ"ב 2022

**תוכן העניינים**

מבוא.................................................................................................................................3

תיאור תחום הידע...............................................................................................................4

מבנה הפרויקט

* תיאור מסך הפתיחה.....................................................................................................5
* תכנון ותיעוד מסכי הפרויקט..........................................................................................6
* תרשים זרימה בין מסכים..............................................................................................9
* תרשים מחלקות (UML).............................................................................................10

מימוש הפרויקט

* מחלקות הפרויקט......................................................................................................11
* בסיס נתונים מקומי (SQLite)......................................................................................12

מדריך למשתמש...............................................................................................................13

סיכום אישי......................................................................................................................15

ביבליוגרפיה.....................................................................................................................16

נספחים............................................................................................................................17

**מבוא**

**תיאור קצר**

המשחק THE IMPOSTOR מבוסס על המשחק האינטרנטי spyfall בו יש קבוצה של אנשים שמקבלים מיקום מסוים (לדוגמא: אולם קולנוע) ומתוך הקבוצה יש שחקן אחד שלא מקבל מיקום - המתחזה.

המטרה של הקבוצה היא לגלות מי המתחזה, המטרה של המתחזה היא לגלות כמה שיותר פרטים על המיקום כדי שלא יחשדו בו. המשתתפים מנסים להשיג את מטרתם בעזרת שאלות שהם מפנים אחד כלפי השני.

האפליקציה תהיה גרסה טלפונית של המשחק ותכלול שדרוגים שהופכים את המשחק ליותר אקטיבי ומעניין.

**קהל היעד**

המשחק מיועד לאוכלוסייה בכל הגילאים.

מומלץ במיוחד לגילאי 8-18 שרוצים להעביר את הזמן עם משחק חברה.

**רקע**

המשחק 'המתחזה' הוא משחק מוכר בצופים, כאשר הגיעה תקופת הקורונה והעברנו את הפעולות בצורה מקוונת היינו צריכים למצוא משחקים מוכרים בגרסה אינטרנטית. בתקופה הזאת הכרתי את spyfall, אני זוכרת שזה היה אחד מהמשחקים האהובים על הקבוצה שלי. גם אחרי שחזרנו לפעילות פרונטלית המשחק עדיין נשאר מוכר ופופולרי.

כשחשבתי על רעיון לפרויקט שלי ידעתי שאני רוצה לעשות משחק חוויתי ומהנה, והדבר הראשון שעלה לי לראש זה אפליקציה שמבוססת על אחד ממשחקים הצופים האהובים עלי – 'המרגל'.

**תיאור תחום הידע**

**שם האובייקט:** Player

**סוג הנתונים:** מייצג את שחקן אחד בקבוצה.

**ייצוג המידע:** מחרוזת עבור השם, משתנה בוליאני עבור האם השחקן הוא המתחזה או לא.

**תיאור פעולות על המידע:** קריאה ועדכון השם, קריאה ועדכון המשתנה הבוליאני.

**שם האובייקט:** Group

**סוג הנתונים:** מייצג קבוצה.

**ייצוג המידע:** מחרוזת עבור שם הקבוצה, מספר שלם עבור מספר השחקנים, מספר שלם עבור מספר המתחזים, מערך של player עבור השחקנים שלך העבודה, מספר שלם עבוד מספר הדקות של המשחק, מחרוזת עבור המיקום.

**תיאור פעולות על המידע:** קריאה ועדכון שם הקבוצה, קריאה ועדכון מספר השחקנים, קריאה ועדכון מספר המתחזים, קריאה ועדכון מספר הדקות, קריאה ועדכון המיקום. איפוס קבוצה (מכניס ערכים למערך של השחקנים, בוחר מיקום).

**מבנה הפרויקט**

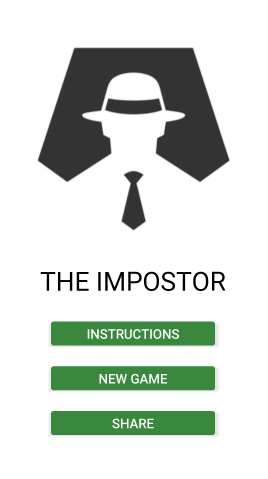
תיאור מסך פתיחה

מסך הפתיחה מכיל את שם המשחק, אייקון, ותפריט דרכו יוכל השחקן לבחור:

לקרוא את ההוראות - יעביר אותו למסך אחר בו כתובות ההוראות.

להתחיל משחק חדש - יפתח דיאלוג אם המשתמש רוצה לשחק עם קבוצה שמורה או חדשה.

לשתף - ישלח הודעה באיזה פלטפורמה שהמשתמש רוצה שמזמינה חבר לשחק.

 תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תכנון ותיעוד מסכי הפרויקט

**מסך הוראות:**

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

המסך גרירה מציג את ההוראות של המשחק.

במסך יש כפתור go back שיחזיר את המשתמש למסך הפתיחה.

**מסך קבוצות שמורות:**

תמונה שמכילה טקסט

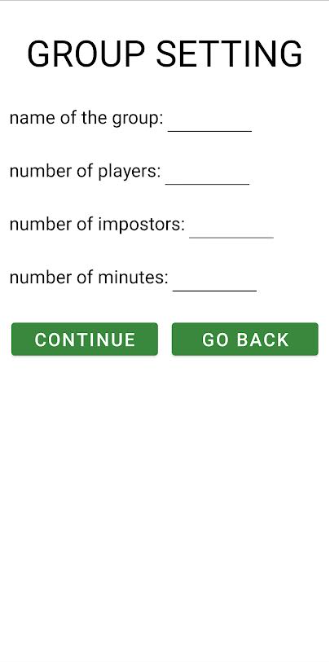
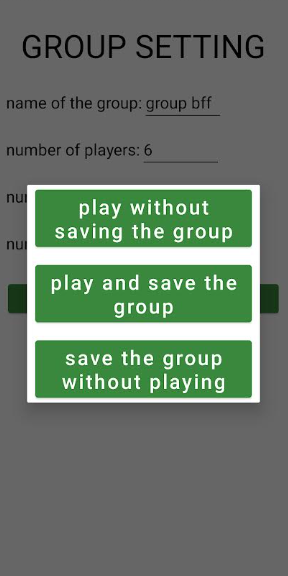
התיאור נוצר באופן אוטומטי

המסך מציג את הקבוצות השמורות בתצוגה של גרירה.

לכל קבוצה יש שלוש אפשרויות - להתחיל משחק (play), למחוק (delete), ולערוך (edit).

במסך יש כפתור go back שיחזיר את המשתמש למסך הפתיחה.

**מסך קבוצה חדשה/ עריכת קבוצה:**

המסך מציג את הגדרות הקבוצה.

במידה והמסך נפתח במטרה לערוך קבוצה שמורה, פרטי הקבוצה יופיעו.

במסך יש כפתור go back שיחזיר את המשתמש למסך הפתיחה.

במסך יש כפתור continue שיפתח דיאולוג שישאל את המשתמש אם הוא רוצה לשמור את הקבוצה ולשחק, לשמור את הקבוצה בלי לשחק, או לשחק בלי לשמור את הקבוצה.

**מסך של שחקן:**



המסך מציג את השם של השחקן והמיקום (אם הוא מתחזה במקום מיקום כתוב לו impostor).

במסך יש כפתור next player שפותח את המסך העברה.

אם השחקן הוא לא המתחזה במסך יש גם כפתור נוסף שמשנה את המיקום של השחקנים (אבל פוסל חצי מרשימת המיקומים של המתחזה).

אם המשחק הוא המתחזה במסך יש שני כפתורים: אחד שפותח רשימת מיקומים שפשר לפסול ממנה, והשני שמבקש רמז – לפסול מיקום (אבל מפעיל התרעה לאחר זמן מה).

**מסך העברה:**

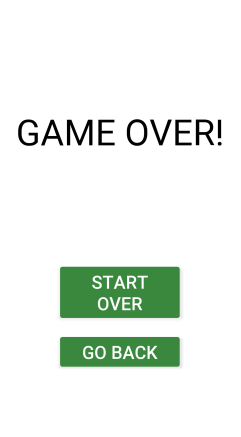
תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

המסך מציג הודעה שצריך להעביר את הטלפון למשתמש מימין ואז לכסות אותו.

כאשר מכסים את הטלפון התאורה שחיישן האור קולט פוחתת ואז המסך שקופץ הוא מסך המשתמש הבא.

**מסך סוף המשחק:**

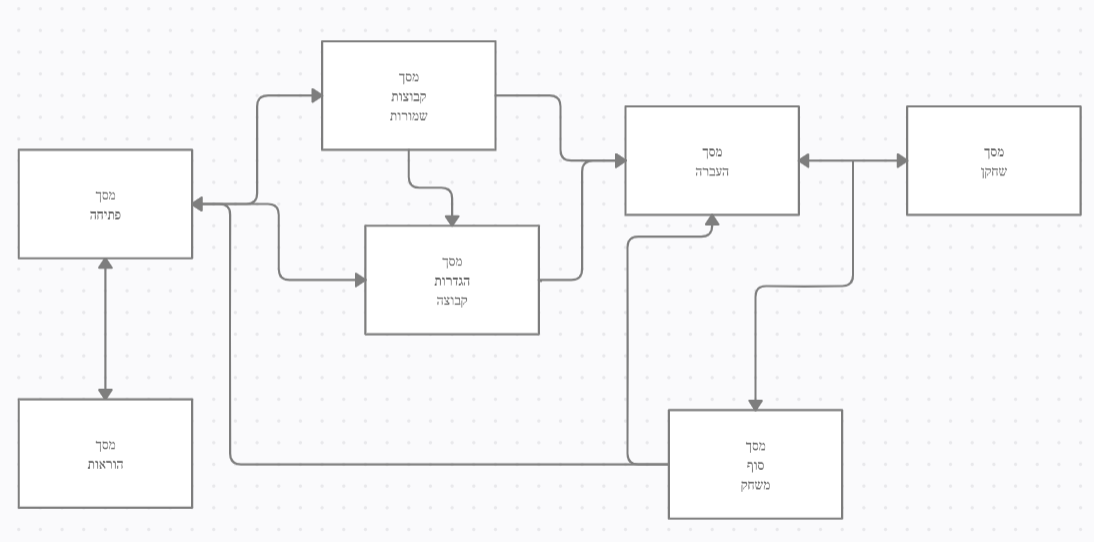


המסך מוצג כאשר הטיימר שמכוון לפי מספק הדקות שהמשתמש הכניס נגמר.

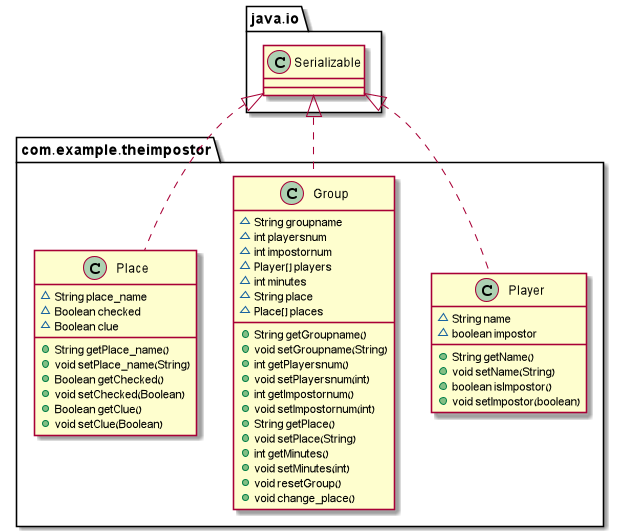
במסך יש כפתור start over שמתחיל את המשחק מחדש (עם אותן הגדרות).

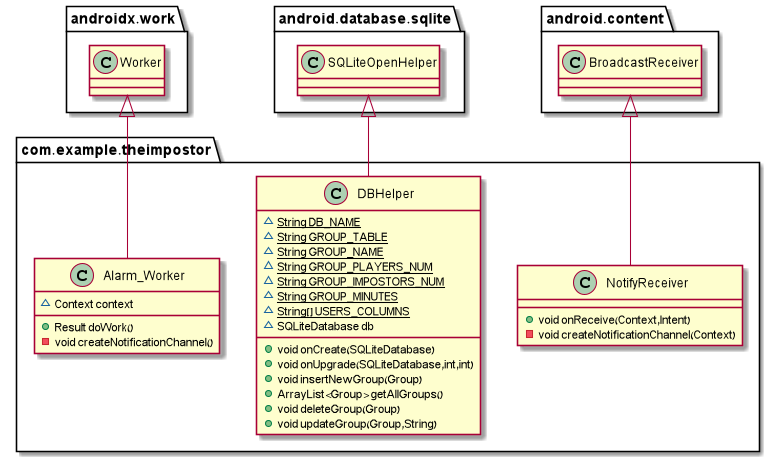
במסך יש כפתור go back שיחזיר את המשתמש למסך הפתיחה.

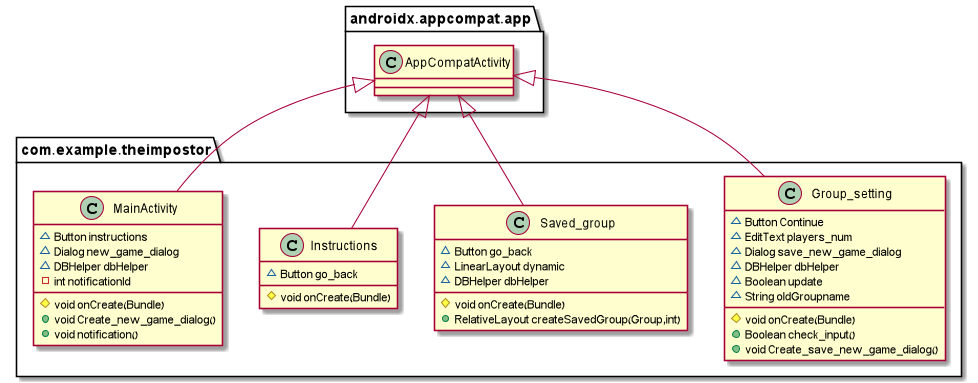
תרשים זרימה בין מסכים

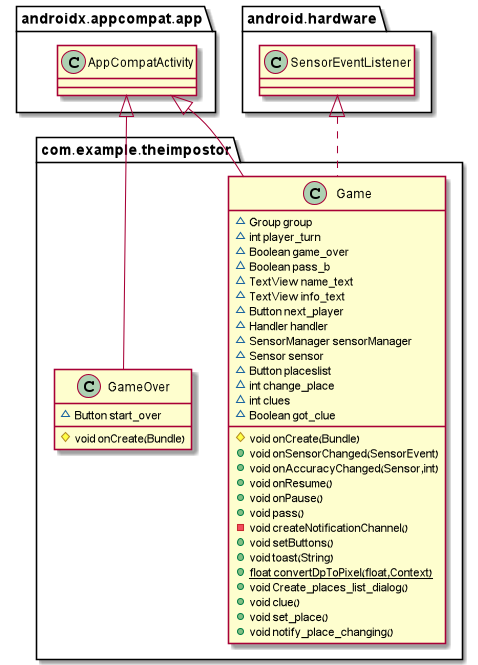


תרשים מחלקות (UML)









**מימוש הפרויקט**

מחלקות הפרויקט

**מחלקת Group**

|  |
| --- |
| תכונות (מאפיינים) |
| String groupname  מחזיק את השם של הקבוצה. |
| int playersnum  מחזיק את מספר השחקנים בקבוצה. |
| int impostornum  מחזיק את מספר המתחזים בקבוצה. |
| Player[] players  מחזיק את המידע על כל השחקנים. |
| int minutes  מחזיק את מספר הדקות של המשחק. |
| String place  מחזיק את המיקום של המשחק. |
| פועלות |
| public void resetGroup ()  מאתחל את הקבוצה: קובע מקום, מחליט מי מבין השחקנים יהיה מתחזה (בהתאם לקלט שהתקבל). |
| change\_place()  משנה את המיקום של השחקנים באמצע המשחק, מאתחל ופוסל חצי מהמיקומים ברשימה של המתחזים. |

**מחלקת Player**

|  |
| --- |
| תכונות (מאפיינים) |
| String name  מחזיק את השם של השחקן. |
| boolean impostor  מחזיק האם השחקן הוא מתחזה או לא. |
| פועלות |

**מחלקת Place**

|  |
| --- |
| תכונות (מאפיינים) |
| String place\_name  מחזיק את השם של המקום. |
| Boolean checked  מחזיק האם המתחזה סימן את המיקום כלא רלוונטי או לא. |
| Boolean clue  מחזיק אם המיקום מפסל בתור רמז או לא. |
| פועלות |

Implicit Intents

**Action**: android.intent.action.SEND

**שימוש**: במסך פתיחה, יש אפשרות לשלוח לחבר הודעה שמזמינה אותו לשחק.

**שם הקובץ בו נמצא**: MainActivity

Service

**שם:** NOTIFICATION\_SERVICE

**סוג:** System Service

**שימוש:** ה-Service משמש ליצירת התראות.

**שם:** ALARM\_SERVICE

**סוג:** System Service

**שימוש:** ה-Service משמש ליצירת התראה לשעה מסויימת.

**שם:** LAYOUT\_INFLATER\_SERVICE

**סוג:** System Service

**שימוש:** ה-Service משמש להצגת הקבוצות השמורות.

**שם:** VIBRATOR\_SERVICE

**סוג:** System Service

**שימוש:** ה-Service משמש להרעדת הטלפון כדי להוגיע למשתמש על המתרחש במשחק.

Broadcast Receivers

**שם:** NotifyReceiver

**אירועים:** action.ACTION\_NEXT\_ALARM\_CLOCK\_CHANGED

**שימוש:** ה-Broadcast מאזין לזמן וכאשר השעה היא 8:00 14:00 או 20:00 קופצת למשתמש התראה שמציעה לו לשחק במשחק.

רכיבים מתקדמים

**סוג הרכיב:** חיישנים

**שימוש:** כאשר מעבירים את הטלפון משחקן לשחקן צריך לכסות את המסך.

**שם הקובץ בו נמצא:** Game

בסיס נתונים מקומי (SQLite)

**שם בסיס הנתונים: MYDB.db**

טבלת קבוצות - gamegroup

הטבלה שומרת את פרטי כל המשתמשים הרשומים באפליקציה.

|  |  |  |
| --- | --- | --- |
| שם עמודה | סוג | תיאור העמודה |
| groupname | text | שם הקבוצה. העמודה מוגדרת כמפתח ראשי. |
| playersnum | integer | מספר השחקנים. |
| impostornum | integer | מספר המתחזים. |
| minutes | integer | מספר הדקות. |

**פעולות על הטבלה:**

הכנסת קבוצה חדשה לטבלה.

שליפת כל הקבוצות מהטבלה.

עדכון הפרטים של קבוצה קיימת.

מחיקת קבוצה.

**מדריך משתמש**

**גרסת אנדרואיד מינימלית**

Android 4.3 API level 16

**מכשירים עליהם נבדקה המערכת**

מכשיר - Samsung Galaxy A70

**הרשאות**

התראות, רטט.

**דרישות מיוחדות ואילוצים**

האפליקציה חייבת חיישן אור.

**אופן הפעולה של הפרויקט**

כאשר פותחים את האפליקציה מופיע מסך הפתיחה, שכולל תפריט עם שלוש אפשרויות:

* לקרוא את ההוראות (יפתח מסך בו כתובות ההוראות).
* לשתף (תשלח הודעה באיזה פלטפורמה שהמשתמש רוצה שמזמינה חבר לשחק).
* להתחיל משחק חדש (יפתח דיאלוג אם המשתמש רוצה לשחק עם קבוצה שמורה או חדשה).

אם המשתמש בחר לשחק עם קבוצה שמורה יפתח מסך בו מופיעות כל הקבוצות השמורות. לכל קבוצה יש שלוש אפשרויות: לערוך, למחוק ולשחק.

אם המשתמש בחר לשחק עם קבוצה חדשה יפתח מסך שבו הוא צריך להכניס את הגדרות הקבוצה:

* שם הקבוצה
* מספר השחקנים (מינימום 3, מקסימום 20)
* מספר המתחזים (מינימום 1, מקסימום שליש ממספר השחקנים)
* מספר הדקות (מינימום 6, מקסימום 12)

לאחר שהמשתמש מכניס את הערכים הרצויים מופיע דיאלוג שמציע לו 3 אפשרויות: לשמור את הקבוצה ולשחק, לשמור את הקבוצה בלי לשחק, או לשחק בלי לשמור את הקבוצה.

במהלך המשחק עצמו יופיע כל פעם מסך אחר שמתאים לשחקן. בין לבין יש מסך מעבר.

בנוסף תופיע התראה שמראה כמה זמן נשאר למשחק, לחיצה עליה תסיים אותו.

לאחר שהזמן נגמר נפתח מסך של סוף המשחק עם שתי אפשרויות: לשחק שוב ולחזור חזרה.

**הודעות למשתמש**

במסך פתיחה:

New game dialog - קבוצה שמורה או קבוצה חדשה?

במסך הגדרות קבוצה:

Continue dialog - לשמור את הקבוצה ולשחק, לשמור את הקבוצה בלי לשחק, או לשחק בלי לשמור את הקבוצה.

Null toast - לא מילאת את כל הפרטים.

Min player toast - מספר השחקנים לא יכול להיות פחות מ3.

Max player toast - מספר השחקנים לא יכול להיות יותר מ20.

Min impostors toast - מספר המתחזים לא יכול להיות פחות מ1.

Max impostors toast - מספר המתחזים לא יכול להיות יותר משליש מספר השחקנים.

Min minutes toast - מספר הדקות לא יכול להיות פחות מ6.

Max minutes toast - מספר הדקות לא יכול להיות יותר מ12.

במשחק עצמו:

Got a clue – להודיע למתחזה שהוא קיבל רמז והספירה לאחור התחילה.

Don’t have clues – להודיע למתחזה שהוא השתמש בכל הרמזים שהיו לו.

Already got a clue – להודיע למתחזה שהוא קיבל כבר רמז בתור הזה והוא צריך לחכות לתור הבא.

Changed the place – להודיע לשחקן שהוא שינה את המיקום ותכף הוא יחשף לכולם.

Already changed the place – להודיע לשחקן שכבר שינו את המיקום במשחק הזה ואי אפשר שוב פעם.

**סיכום אישי**

בתחילת השנה כשהתחלנו ללמוד על אנדרואיד סטודיו ולהכיר יותר טוב את השפה java, הפרויקט היה נראה כמו משימה קשה, מסובכת וארוכה (על גבול הבלתי אפשרית). אך עם הזמן הידע שלי התפתח והורחב והבנתי שבעזרת ניהול זמן נכון והמון מחויבות ומוכנות, אפשר להפוך את הפרויקט הזה לחוויה ולצלוח אותו.

בהתחלה הייתי צריכה לחשוב על נושא, במהלך השנה היו לי מספר רעיונות. אך בסופו של דבר הרעיון שבחרתי היה המשחק 'המתחזה', מאוד התחברתי למשחק כי הוא קשור לצופים (שתופסים חלק גדול מהחיים שלי).

לאחר שהיה לי רעיון, נעזרתי במורה שלי, רועי אמיגה וביחד חשבנו איך אפשר לשפר את המשחק (שיענה על כל הדרישות) ואיך אפשר להפוך את הרעיון המופשט לקוד ממשי. ואז התחלתי לכתוב את הקוד לאפליקציה.

מבחינת הרמה של האפליקציה והמימוש שלה אני מאמינה שהגעתי ליעד שלי. אני מאוד מרוצה מהתוצר הסופי של הפרויקט, אני מרגישה שהוא מציג בצורה מצוינת את המאמצים הרבים שהשקעתי בעבודה. יעד נוסף שהיה לי, הוא להנות וללמוד במהלך בתהליך יצירת האפליקציה, גם הידע הזה הושג - היה מעניין לחקור על סביבת העבודה, התעמקתי בנושאים רבים ולמדתי המון דברים חדשים שלא ידעתי קודם לכן.

היו קשיים רבים שעלו במהלך הכנת הפרויקט, חלקם נפתרו בקלות, אך חלקם דרשו יותר מאמץ וזמן כדי לפתור אותם ובכך גם יצרו תסכול לא קטן. בסופו של דבר, את כל הבעיות שעלו במהלך כתיבת הקוד הצלחתי לפתור בצורה כזו או אחרת. ותחושת ההצלחה היא זאת שמשכה אותי קדימה – לעבוד ולהשקיע.

קושי נוסף שעלה הוא הניהול של הזמן – היה קשה לשלב את הכנת הפרויקט עם הלוז העמוס שלי שכולל צופים עם לימודי פיזיקה ומתמטיקה, אך היה לי חשוב להצליח ולהספיק הכל. אז עם סדרי עדיפויות וניהול זמן נכון הצלחתי להתגבר על הקושי.

בסופו של דבר אני מרגישה שהפרויקט תרם לי המון כלים שאוכל להשתמש בהם בעתיד גם בהקשר של תכנות וגם מעבר. התנסיתי בפתרון בעיות בצורה עצמאית בעזרת חיפוש מידע רלוונטי באינטרנט, אז היכולת שלי לאתר מידע השתפרה וגם השליטה שלי בשפה האנגלית (בה ביצעתי את החיפושים). בנוסף, למדתי על ניהול זמן, על הובלת פרויקט, על התמודדות עם קשיים, על התנהלות מול מלוווים (המורה שלי – רועי אמיגה), וכו'...

אני חושבת שהדבר העיקרי שאני לוקחת מהעבודה הזאת זה את תחושת המסוגלות שקיבלתי לאחר שלקחתי רעיון מופשט לגמרי והצלחתי להפוך אותו למציאות, זה משהו בלתי נתפס בעיני, ואני מאוד שמחה שיצא לי להתנסות בזה.

לסיכום, אני מאוד מודה על ההזדמנות לקחת חלק במגמה המדהימה הזו ולסיים שלוש שנים עם פרויקט מיוחד שמבטא בצורה מצוינת את ההשקעה והעבודה הקשה שלי. וגם אני מודה על כל הידע, כלים ויכולות, שרכשתי בתהליך.

**נספחים**

public class Alarm\_Worker extends Worker {  
 Context context;  
  
 public Alarm\_Worker(@NonNull Context context, @NonNull WorkerParameters workerParams) {  
 super(context, workerParams);  
 this.context = context;  
 }  
  
 @NonNull  
 @Override  
 public Result doWork() {  
 NotificationCompat.Builder notification = new NotificationCompat.Builder(context, "id");  
 notification.setContentTitle("THE IMPOSTOR GOT A CLUE");  
 notification.setSmallIcon(R.drawable.*impostor*);  
 notification.setWhen(System.*currentTimeMillis*());  
 notification.setAutoCancel(true);  
 NotificationManager notificationManager = (NotificationManager)context.getSystemService(Context.*NOTIFICATION\_SERVICE*);  
 notificationManager.notify(3, notification.build());  
  
 Vibrator v = (Vibrator) context.getSystemService(Context.*VIBRATOR\_SERVICE*);  
 *// Vibrate for 500 milliseconds* if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 v.vibrate(VibrationEffect.*createOneShot*(5000, VibrationEffect.*DEFAULT\_AMPLITUDE*));  
 } else {  
 *//deprecated in API 26* v.vibrate(6700);  
 }  
 MediaPlayer alarm = MediaPlayer.*create*(context ,R.raw.*alarm*);  
 alarm.start();  
  
 return Result.*success*();  
 }  
  
 private void createNotificationChannel() {  
 *// Create the NotificationChannel, but only on API 26+ because  
 // the NotificationChannel class is new and not in the support library* if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 CharSequence name = "Notifications";  
 String description = "this is notifications";  
 int importance = NotificationManager.*IMPORTANCE\_DEFAULT*;  
 NotificationChannel channel = new NotificationChannel("id", name, importance);  
 channel.setDescription(description);  
 *// Register the channel with the system; you can't change the importance  
 // or other notification behaviors after this* NotificationManager notificationManager = context.getSystemService(NotificationManager.class);  
 notificationManager.createNotificationChannel(channel);  
 }  
 }  
}

public class DBHelper extends SQLiteOpenHelper {  
  
 final static String *DB\_NAME* = "MYDB.db";  
 final static String *GROUP\_TABLE* = "gamegroup";  
 final static String *GROUP\_NAME* = "groupname";  
 final static String *GROUP\_PLAYERS\_NUM* = "playersnum";  
 final static String *GROUP\_IMPOSTORS\_NUM* = "impostornum";  
 final static String *GROUP\_MINUTES* = "minutes";  
 final static String[] *USERS\_COLUMNS* = {*GROUP\_NAME*, *GROUP\_PLAYERS\_NUM*, *GROUP\_IMPOSTORS\_NUM*, *GROUP\_MINUTES*};  
 SQLiteDatabase db;  
  
 public DBHelper(@Nullable Context context) {  
 super(context, *DB\_NAME*, null, 1);  
 db = getWritableDatabase();  
 }  
  
 @Override  
 public void onCreate(SQLiteDatabase db) {  
 String createGroupTableQuery =  
 "create table " + *GROUP\_TABLE* +  
 " (" + *GROUP\_NAME* + " text," +  
 " " + *GROUP\_PLAYERS\_NUM* + " integer," +  
 " " + *GROUP\_IMPOSTORS\_NUM* + " integer," +  
 " " + *GROUP\_MINUTES* + " integer);";  
  
 Log.*d*("DBTEST", createGroupTableQuery);  
 db.execSQL(createGroupTableQuery);  
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
 db.execSQL("DROP TABLE IF EXISTS " + *GROUP\_NAME*);  
 onCreate(db);  
 }  
  
 public void insertNewGroup (Group group){  
 db = getWritableDatabase();  
 ContentValues contentValues = new ContentValues();  
 contentValues.put(*GROUP\_NAME*, group.getGroupname());  
 contentValues.put(*GROUP\_PLAYERS\_NUM*, group.getPlayersnum());  
 contentValues.put(*GROUP\_IMPOSTORS\_NUM*, group.getImpostornum());  
 contentValues.put(*GROUP\_MINUTES*, group.getMinutes());  
 db.insert(*GROUP\_TABLE*, null, contentValues);  
 }  
  
 public ArrayList<Group> getAllGroups()  
 {  
 ArrayList<Group> groups = new ArrayList<>();  
 db = getReadableDatabase();  
 Cursor cursor = db.query(*GROUP\_TABLE*, null, null,  
 null, null, null, null);  
 cursor.moveToFirst();  
 while(!cursor.isAfterLast())  
 {  
 String Groupname = cursor.getString(  
 (int) cursor.getColumnIndex(*GROUP\_NAME*));  
 int Playersnum = cursor.getInt(  
 (int)cursor.getColumnIndex(*GROUP\_PLAYERS\_NUM*));  
 int Impostornum = cursor.getInt(  
 (int)cursor.getColumnIndex(*GROUP\_IMPOSTORS\_NUM*));  
 int minutes = cursor.getInt(  
 (int)cursor.getColumnIndex(*GROUP\_MINUTES*));  
  
 Group group = new Group(Groupname, Playersnum, Impostornum, minutes);  
 groups.add(group);  
 cursor.moveToNext();  
 }  
 return groups;  
 }  
  
 public void deleteGroup(Group group)  
 {  
 SQLiteDatabase db = getWritableDatabase();  
 db.delete(*GROUP\_TABLE*, *GROUP\_NAME* + " = '" + group.getGroupname() + "'", null);  
 }  
  
 public void updateGroup(Group group, String oldGroupname)  
 {  
 SQLiteDatabase db = getWritableDatabase();  
 ContentValues contentValues = new ContentValues();  
 contentValues.put(*GROUP\_NAME*, group.getGroupname());  
 contentValues.put(*GROUP\_PLAYERS\_NUM*, group.getPlayersnum());  
 contentValues.put(*GROUP\_IMPOSTORS\_NUM*, group.getImpostornum());  
 contentValues.put(*GROUP\_MINUTES*, group.getMinutes());  
 db.update(*GROUP\_TABLE*, contentValues, *GROUP\_NAME* + " = '" + oldGroupname + "'", null);  
 }  
}

public class Game extends AppCompatActivity implements SensorEventListener {  
  
 Group group;  
 int player\_turn;  
 Boolean game\_over;  
 Boolean pass\_b, next\_player\_b;  
 TextView name\_text;  
 TextView info\_text;  
 Button next\_player;  
 Handler handler = new Handler();  
 SensorManager sensorManager;  
 Sensor sensor;  
 Button placeslist, clue, changeplace;  
 int change\_place = 1;  
 int clues = 2;  
 Boolean got\_clue;  
  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*pass\_phone\_layout*);  
  
 getSupportActionBar().hide();  
  
 Window window = this.getWindow();  
 window.addFlags(WindowManager.LayoutParams.*FLAG\_DRAWS\_SYSTEM\_BAR\_BACKGROUNDS*);  
 window.clearFlags(WindowManager.LayoutParams.*FLAG\_TRANSLUCENT\_STATUS*);  
 window.setStatusBarColor(this.getResources().getColor(R.color.*black*));  
  
 Intent intent = getIntent();  
 group = (Group)intent.getSerializableExtra("group");  
  
 game\_over = false;  
 player\_turn = 0;  
 sensorManager = (SensorManager) getSystemService(Context.*SENSOR\_SERVICE*);  
 sensor = sensorManager.getDefaultSensor(Sensor.*TYPE\_LIGHT*);  
 setButtons();  
  
 Thread myThread = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 while (game\_over != true){  
 pass();  
 if (player\_turn < group.players.length-1){  
 player\_turn++;  
 }  
 else {  
 player\_turn = 0;  
 }  
 }  
 }  
 });  
 myThread.start();  
  
 Intent intent1 = new Intent(this, MainActivity.class);  
 intent1.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK* | Intent.*FLAG\_ACTIVITY\_CLEAR\_TASK*);  
 PendingIntent pendingIntent1 = PendingIntent.*getActivity*(this, 0, intent1, 0);  
  
 createNotificationChannel();  
 NotificationCompat.Builder notification = new NotificationCompat.Builder(this, "id");  
 notification.setContentTitle("The game is on!");  
 notification.setContentText("time left: " + group.minutes + ":00 | click to end the game" );  
 notification.setSmallIcon(R.drawable.*impostor*);  
 notification.setPriority(Notification.*PRIORITY\_DEFAULT*);  
 notification.setContentIntent(pendingIntent1);  
 notification.setAutoCancel(true);  
  
 NotificationManager notificationManager = (NotificationManager)getSystemService(Context.*NOTIFICATION\_SERVICE*);  
 notificationManager.notify(2, notification.build());  
  
 new CountDownTimer((int)group.getMinutes()\*1000\*60, 1000){  
 public void onTick(long millisUntilFinished){  
 int minutes\_left = (int) millisUntilFinished / 60000;  
 int seconds\_left = (int) (millisUntilFinished - minutes\_left\*60000)/1000;  
 notification.setContentText("time left: " + minutes\_left + ":" + seconds\_left + " | click to end the game" );  
 notificationManager.notify(2, notification.build());  
 }  
 public void onFinish(){  
 notificationManager.cancel(2);  
 game\_over = true;  
 Intent intent = new Intent(Game.this, GameOver.class);  
 intent.putExtra("group", group);  
 startActivity(intent);  
 }  
 }.start();  
 }  
  
 @Override  
 public void onSensorChanged(SensorEvent event) {  
 float currentSensorValue = event.values[0];  
 if (currentSensorValue <= 50){  
 pass\_b = true;  
 }  
 }  
  
 @Override  
 public void onAccuracyChanged(Sensor sensor, int accuracy) {  
  
 }  
  
 @Override  
 public void onResume(){  
 super.onResume();  
 sensorManager.registerListener(this, sensor, SensorManager.*SENSOR\_DELAY\_NORMAL*);  
 }  
  
 @Override  
 public void onPause(){  
 super.onPause();  
 sensorManager.unregisterListener(this);  
 }  
  
 public void pass(){  
 handler.post(new Runnable() {  
 @Override  
 public void run() {  
 setContentView(R.layout.*pass\_phone\_layout*);  
 }  
 });  
  
 pass\_b = false;  
  
 while (pass\_b != true){  
 *//wait* }  
  
 got\_clue = true;  
  
 handler.post(new Runnable() {  
 @Override  
 public void run() {  
 setContentView(R.layout.*player\_layout*);  
 name\_text = findViewById(R.id.*name\_text*);  
 info\_text = findViewById(R.id.*info\_text*);  
 next\_player = findViewById(R.id.*next\_player*);  
 name\_text.setText(group.players[player\_turn].getName());  
 LinearLayout layout = findViewById(R.id.*buttons\_layout*);  
  
 if(changeplace.getParent() != null) {  
 ((ViewGroup)changeplace.getParent()).removeView(changeplace);  
 }  
 if(placeslist.getParent() != null) {  
 ((ViewGroup)placeslist.getParent()).removeView(placeslist);  
 }  
 if(clue.getParent() != null) {  
 ((ViewGroup)clue.getParent()).removeView(clue);  
 }  
  
 if (group.players[player\_turn].isImpostor() == true){  
 info\_text.setText("impostor");  
 layout.addView(placeslist);  
 layout.addView(clue);  
 }  
 else{  
 info\_text.setText(group.getPlace());  
 layout.addView(changeplace);  
 }  
 }  
 });  
  
 next\_player\_b = false;  
  
 handler.post(new Runnable() {  
 @Override  
 public void run() {  
 next\_player.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 next\_player\_b = true;  
 }  
 }); }  
 });  
  
  
 while (next\_player\_b != true) {  
 *// wait* }  
 }  
  
 private void createNotificationChannel () {  
 *// Create the NotificationChannel, but only on API 26+ because  
 // the NotificationChannel class is new and not in the support library* if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 CharSequence name = "Notifications";  
 String description = "this is notifications";  
 int importance = NotificationManager.*IMPORTANCE\_DEFAULT*;  
 NotificationChannel channel = new NotificationChannel("id", name, importance);  
 channel.setDescription(description);  
 *// Register the channel with the system; you can't change the importance  
 // or other notification behaviors after this* NotificationManager notificationManager = getSystemService(NotificationManager.class);  
 notificationManager.createNotificationChannel(channel);  
 }  
 }  
  
 public void setButtons(){  
 Random random = new Random();  
 WorkManager workManager = WorkManager.*getInstance*(this);  
 LinearLayout.LayoutParams params = new LinearLayout.LayoutParams((int) *convertDpToPixel*(300, this), LinearLayout.LayoutParams.*WRAP\_CONTENT*);  
 params.setMargins(0, 20, 0, 0);  
  
 placeslist = new Button(this);  
 placeslist.setLayoutParams(params);  
 placeslist.setText("list of places");  
 placeslist.setBackgroundResource(R.drawable.*green\_color*);  
 placeslist.setTextSize(25);  
 placeslist.setTextColor(Color.*WHITE*);  
  
 placeslist.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Create\_places\_list\_dialog();  
 }  
 });  
  
 clue = new Button(this);  
 clue.setLayoutParams(params);  
 clue.setText("get clue");  
 clue.setBackgroundResource(R.drawable.*green\_color*);  
 clue.setTextSize(25);  
 clue.setTextColor(Color.*WHITE*);  
  
 clue.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (clues != 0 || got\_clue == false){  
 clues --;  
 clue();  
 got\_clue = true;  
 Data.Builder data = new Data.Builder();  
 data.putString("type", "clue");  
 OneTimeWorkRequest oneTimeWorkRequest =  
 new OneTimeWorkRequest.Builder(Alarm\_Worker.class)  
 .setConstraints(Constraints.*NONE*)  
 .setInitialDelay((random.nextInt(20)+20)\*1000, TimeUnit.*SECONDS*)  
 .addTag("clue")  
 .setInputData(data.build())  
 .build();  
 workManager.enqueue(oneTimeWorkRequest);  
 toast("You got a clue!");  
 }  
 else {  
 if (clues != 0) {  
 toast("You dont have any clues left");  
 }  
 else {  
 toast("You already got a clue this turn");  
 }  
 }  
 }  
 });  
  
 changeplace = new Button(this);  
 changeplace.setLayoutParams(params);  
 changeplace.setText("change place");  
 changeplace.setBackgroundResource(R.drawable.*green\_color*);  
 changeplace.setTextSize(25);  
 changeplace.setTextColor(Color.*WHITE*);  
  
 changeplace.setOnClickListener(new View.OnClickListener() {  
 @SuppressLint("RestrictedApi")  
 @Override  
 public void onClick(View v) {  
 if (change\_place != 0) {  
 change\_place --;  
 set\_place();  
 toast("You changed the place!");  
 }  
 else {  
 toast("You already changed the place");  
 }  
 }  
 });  
 }  
  
 public void toast(String string){  
 Toast mytoast = new Toast(this);  
 mytoast.setText(string);  
 mytoast.setDuration(Toast.*LENGTH\_SHORT*);  
 mytoast.show();  
 }  
  
 public static float convertDpToPixel(float dp, Context context){  
 return dp \* ((float) context.getResources().getDisplayMetrics().densityDpi / DisplayMetrics.*DENSITY\_DEFAULT*);  
 }  
  
 public void Create\_places\_list\_dialog() {  
 Dialog places\_list\_dialog = new Dialog(this);  
 LinearLayout places\_list\_layout = new LinearLayout(this);  
 LinearLayout.LayoutParams params =  
 new LinearLayout.LayoutParams(LinearLayout.LayoutParams.*WRAP\_CONTENT*, LinearLayout.LayoutParams.*MATCH\_PARENT*);  
 places\_list\_layout.setLayoutParams(params);  
 places\_list\_layout.setOrientation(LinearLayout.*VERTICAL*);  
  
 for (int i = 0; i < group.places.length; i++) {  
 TextView textView = new TextView(this);  
 textView.setText(group.places[i].getPlace\_name());  
 textView.setTextColor(Color.*BLACK*);  
 textView.setTextSize(15);  
 if (group.places[i].getChecked() == true || group.places[i].getClue() == true) {  
 textView.setPaintFlags(textView.getPaintFlags() | Paint.*STRIKE\_THRU\_TEXT\_FLAG*);  
 }  
 textView.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 TextView textView = (TextView) v;  
 int place\_i = -1;  
 for (int i = 0; i < group.places.length; i++) {  
 if (group.places[i].getPlace\_name().equals(((TextView) v).getText().toString())){  
 place\_i = i;  
 }  
 }  
 if (place\_i != -1) {  
 if (group.places[place\_i].getChecked() == true) {  
 textView.setPaintFlags(textView.getPaintFlags() & (~Paint.*STRIKE\_THRU\_TEXT\_FLAG*));  
 group.places[place\_i].setChecked(false);  
 }  
 else {  
 textView.setPaintFlags(textView.getPaintFlags() | Paint.*STRIKE\_THRU\_TEXT\_FLAG*);  
 group.places[place\_i].setChecked(true);  
 }  
 }  
 }  
 });  
 places\_list\_layout.addView(textView);  
 }  
 places\_list\_dialog.setContentView(places\_list\_layout);  
 places\_list\_dialog.setTitle("PLACES LIST");  
 places\_list\_dialog.setCancelable(true);  
 places\_list\_dialog.show();  
 }  
  
 public void clue(){  
 Random random = new Random();  
 int clues\_num = 0;  
 int clues\_possible\_num = group.places.length + 1;  
 for (int i = 0 ; i < group.places.length ; i ++)  
 {  
 if (group.places[i].getChecked() == true || group.places[i].getClue() == true) {  
 clues\_possible\_num --;  
 }  
 }  
  
 Boolean find\_clues\_num = true;  
 while (find\_clues\_num == true){  
 clues\_num = random.nextInt(3) + 1;  
 if (clues\_num < clues\_possible\_num){  
 find\_clues\_num = false;  
 }  
 }  
  
 while (clues\_num != 0) {  
 int i = random.nextInt(20);  
 if (group.places[i].getChecked() == false && group.places[i].getClue() == false && group.places[i].getPlace\_name().equals(group.place) != true) {  
 group.places[i].setClue(true);  
 clues\_num --;  
 }  
 }  
 }  
  
 public void set\_place(){  
 Random random = new Random();  
 new CountDownTimer((random.nextInt(20)+20)\*1000, 100){  
 public void onTick(long millisUntilFinished){  
 }  
 public void onFinish(){  
 notify\_place\_changing();  
 group.change\_place();  
 }  
 }.start();  
 }  
  
 public void notify\_place\_changing(){  
 NotificationCompat.Builder notification = new NotificationCompat.Builder(this, "id");  
 notification.setContentTitle("THE PLACE HAS CHANGED");  
 notification.setContentText("please pass the phone a full round");  
 notification.setSmallIcon(R.drawable.*impostor*);  
 notification.setWhen(System.*currentTimeMillis*());  
 notification.setAutoCancel(true);  
 NotificationManager notificationManager = (NotificationManager)getSystemService(Context.*NOTIFICATION\_SERVICE*);  
 notificationManager.notify(4, notification.build());  
  
 Vibrator v = (Vibrator) getSystemService(Context.*VIBRATOR\_SERVICE*);  
 *// Vibrate for 500 milliseconds* if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 v.vibrate(VibrationEffect.*createOneShot*(5000, VibrationEffect.*DEFAULT\_AMPLITUDE*));  
 } else {  
 *//deprecated in API 26* v.vibrate(6700);  
 }  
 MediaPlayer alarm = MediaPlayer.*create*(this ,R.raw.*alarm*);  
 alarm.start();  
 }  
}

public class GameOver extends AppCompatActivity {  
  
 Button start\_over, go\_back;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_game\_over*);  
  
 getSupportActionBar().hide();  
  
 Window window = this.getWindow();  
 window.addFlags(WindowManager.LayoutParams.*FLAG\_DRAWS\_SYSTEM\_BAR\_BACKGROUNDS*);  
 window.clearFlags(WindowManager.LayoutParams.*FLAG\_TRANSLUCENT\_STATUS*);  
 window.setStatusBarColor(this.getResources().getColor(R.color.*black*));  
  
 start\_over = findViewById(R.id.*start\_over*);  
 go\_back = findViewById(R.id.*go\_back*);  
  
 go\_back.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent(GameOver.this, MainActivity.class);  
 startActivity(intent);  
 }  
 });  
  
 start\_over.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent group\_intent = getIntent();  
 Group group = (Group)group\_intent.getSerializableExtra("group");  
 group.resetGroup();  
  
 Intent intent = new Intent(GameOver.this, Game.class);  
 intent.putExtra("group", group);  
 startActivity(intent);  
 }  
 });  
 }  
}

public class Group implements Serializable {  
  
 String groupname;  
 int playersnum;  
 int impostornum;  
 Player[] players;  
 int minutes;  
 String place;  
 Place[] places;  
  
 public Group (String groupname, int playersnum, int impostornum, int minutes){  
 this.groupname = groupname;  
 this.playersnum = playersnum;  
 this.impostornum = impostornum;  
 this.players = new Player[playersnum];  
 this.minutes = minutes;  
 }  
  
 public String getGroupname() { return groupname; }  
  
 public void setGroupname(String groupname) { this.groupname = groupname; }  
  
 public int getPlayersnum() { return playersnum; }  
  
 public void setPlayersnum(int playersnum) { this.playersnum = playersnum; }  
  
 public int getImpostornum() { return impostornum; }  
  
 public void setImpostornum(int impostornum) { this.impostornum = impostornum; }  
  
 public String getPlace() { return place; }  
  
 public void setPlace(String place) { this.place = place; }  
  
 public int getMinutes() { return minutes; }  
  
 public void setMinutes(int minutes) { this.minutes = minutes; }  
  
 public void resetGroup (){  
 Random random = new Random();  
 this.places = new Place[]{new Place("Airplane"), new Place("Bank"), new Place("Casino"), new Place("Circus Tent"), new Place("Party"), new Place("Theater"), new Place("Spa"), new Place("Hospital"), new Place("Hotel"), new Place("Military Base"), new Place("Movie"), new Place("Train"), new Place("Pirate Ship"), new Place("Space Station"), new Place("Police Station"), new Place("Restaurant"), new Place("School"), new Place("Supermarket"), new Place("Zoo"), new Place("Library")};  
 place = places[random.nextInt(places.length)].getPlace\_name();  
  
 for(int i=0 ; i<players.length ; i++) {  
 players[i] = new Player();  
 players[i].setName("player " + (i+1));  
 players[i].setImpostor(false);  
 }  
  
 int[] impostors = new int[this.impostornum];  
 Boolean duplicate = true;  
 while (duplicate){  
 for(int i=0 ; i<impostors.length ; i++) {  
 impostors[i] = random.nextInt(playersnum);  
 }  
 duplicate = false;  
 for(int i = 0; i < impostors.length; i++){  
 for(int j = i + 1; j < impostors.length; j++){  
 if(impostors[i] == impostors[j]){  
 duplicate = true;  
 }  
 }  
 }  
 }  
 for(int i=0 ; i<impostors.length ; i++) {  
 players[impostors[i]].setImpostor(true);  
 }  
 }  
  
 public void change\_place() {  
 Random random = new Random();  
 boolean same = true;  
 while (same) {  
 String new\_place = places[random.nextInt(places.length)].getPlace\_name();  
 if (place.equals(new\_place) != true) {  
 same = false;  
 place = new\_place;  
 }  
 }  
  
 int clues\_num = 10;  
 while (clues\_num != 0) {  
 int i = random.nextInt(20);  
 if (places[i].getPlace\_name().equals(place) != true) {  
 places[i].setClue(true);  
 clues\_num --;  
 }  
 }  
 }  
}

public class Group\_setting extends AppCompatActivity {  
  
 Button Continue, go\_back, play\_dont\_save, play\_save, dont\_play\_save;  
 EditText players\_num, impostor\_num, group\_name, minutes;  
 Dialog save\_new\_game\_dialog;  
 DBHelper dbHelper;  
 Boolean update = false;  
 String oldGroupname;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_group\_setting*);  
  
 getSupportActionBar().hide();  
  
 Window window = this.getWindow();  
 window.addFlags(WindowManager.LayoutParams.*FLAG\_DRAWS\_SYSTEM\_BAR\_BACKGROUNDS*);  
 window.clearFlags(WindowManager.LayoutParams.*FLAG\_TRANSLUCENT\_STATUS*);  
 window.setStatusBarColor(this.getResources().getColor(R.color.*black*));  
  
 dbHelper = new DBHelper(this);  
  
 players\_num = findViewById(R.id.*players\_num*);  
 impostor\_num = findViewById(R.id.*impostor\_num*);  
 group\_name = findViewById(R.id.*group\_name*);  
 minutes = findViewById(R.id.*minutes*);  
 Continue = findViewById(R.id.*Continue*);  
 go\_back = findViewById(R.id.*go\_back*);  
  
  
 Intent intent = getIntent();  
 if (intent.hasExtra("i")) {  
 update = true;  
 ArrayList<Group> groups = new ArrayList<>();  
 groups = dbHelper.getAllGroups();  
 Group group = groups.get(Integer.*parseInt*(intent.getExtras().getString("i")));  
 players\_num.setText(Integer.*toString*(group.getPlayersnum()));  
 impostor\_num.setText(Integer.*toString*(group.getImpostornum()));  
 minutes.setText(Integer.*toString*(group.getMinutes()));  
 oldGroupname = group.getGroupname();  
 group\_name.setText(oldGroupname);  
 }  
  
 Continue.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (check\_input()){  
 Create\_save\_new\_game\_dialog();  
 }  
 }  
 });  
  
 go\_back.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent(Group\_setting.this, MainActivity.class);  
 startActivity(intent);  
 }  
 });  
  
 }  
  
 public Boolean check\_input (){  
 if (group\_name.getText().toString().matches("") || players\_num.getText().toString().matches("") || impostor\_num.getText().toString().matches("") || minutes.getText().toString().matches("")){  
 Toast mytoast = new Toast(this);  
 mytoast.setText("You did not enter all the details");  
 mytoast.setDuration(Toast.*LENGTH\_SHORT*);  
 mytoast.show();  
 return false;  
 }  
 if (Integer.*parseInt*(players\_num.getText().toString())<3){  
 Toast mytoast = new Toast(this);  
 mytoast.setText("number of players cannot be less than 3");  
 mytoast.setDuration(Toast.*LENGTH\_SHORT*);  
 mytoast.show();  
 return false;  
 }  
 if (Integer.*parseInt*(players\_num.getText().toString())>20){  
 Toast mytoast = new Toast(this);  
 mytoast.setText("number of players cannot be more than 20");  
 mytoast.setDuration(Toast.*LENGTH\_SHORT*);  
 mytoast.show();  
 return false;  
 }  
 if (Integer.*parseInt*(impostor\_num.getText().toString())<1){  
 Toast mytoast = new Toast(this);  
 mytoast.setText("There should be at least 1 impostor");  
 mytoast.setDuration(Toast.*LENGTH\_SHORT*);  
 mytoast.show();  
 return false;  
 }  
 if (Integer.*parseInt*(impostor\_num.getText().toString()) > Integer.*parseInt*(players\_num.getText().toString())/3){  
 Toast mytoast = new Toast(this);  
 mytoast.setText("number of impostors cannot be more than " + Integer.*parseInt*(players\_num.getText().toString())/3);  
 mytoast.setDuration(Toast.*LENGTH\_SHORT*);  
 mytoast.show();  
 return false;  
 }  
 if (Integer.*parseInt*(minutes.getText().toString()) <6){  
 Toast mytoast = new Toast(this);  
 mytoast.setText("number of minutes cannot be less than 6");  
 mytoast.setDuration(Toast.*LENGTH\_SHORT*);  
 mytoast.show();  
 return false;  
 }  
 if (Integer.*parseInt*(minutes.getText().toString()) > 12){  
 Toast mytoast = new Toast(this);  
 mytoast.setText("number of minutes cannot be more than 12");  
 mytoast.setDuration(Toast.*LENGTH\_SHORT*);  
 mytoast.show();  
 return false;  
 }  
 return true;  
 }  
  
 public void Create\_save\_new\_game\_dialog(){  
 save\_new\_game\_dialog = new Dialog(this);  
 save\_new\_game\_dialog.setContentView(R.layout.*continue\_layout*);  
 save\_new\_game\_dialog.setTitle("");  
 save\_new\_game\_dialog.setCancelable(true);  
 play\_dont\_save = save\_new\_game\_dialog.findViewById(R.id.*play\_dont\_save*);  
 play\_save = save\_new\_game\_dialog.findViewById(R.id.*play\_save*);  
 dont\_play\_save = save\_new\_game\_dialog.findViewById(R.id.*dont\_play\_save*);  
  
 play\_save.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Group group = new Group(group\_name.getText().toString(), Integer.*parseInt*(players\_num.getText().toString()), Integer.*parseInt*(impostor\_num.getText().toString()), Integer.*parseInt*(minutes.getText().toString()));  
 if (update){  
 dbHelper.updateGroup(group, oldGroupname);  
 }  
 else {  
 dbHelper.insertNewGroup(group);  
 }  
 group.resetGroup();  
 Intent intent = new Intent(Group\_setting.this, Game.class);  
 intent.putExtra("group", group);  
 startActivity(intent);  
 *// start game :)* }  
 });  
  
 play\_dont\_save.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Group group = new Group(group\_name.getText().toString(), Integer.*parseInt*(players\_num.getText().toString()), Integer.*parseInt*(impostor\_num.getText().toString()), Integer.*parseInt*(minutes.getText().toString()));  
 group.resetGroup();  
 Intent intent = new Intent(Group\_setting.this, Game.class);  
 intent.putExtra("group", group);  
 startActivity(intent);  
 *// start game :)* }  
 });  
  
 dont\_play\_save.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Group group = new Group(group\_name.getText().toString(), Integer.*parseInt*(players\_num.getText().toString()), Integer.*parseInt*(impostor\_num.getText().toString()), Integer.*parseInt*(minutes.getText().toString()));  
 if (update){  
 dbHelper.updateGroup(group, oldGroupname);  
 }  
 else {  
 dbHelper.insertNewGroup(group);  
 }  
 Intent intent = new Intent(Group\_setting.this, MainActivity.class);  
 startActivity(intent);  
 }  
 });  
  
 save\_new\_game\_dialog.show();  
 }  
}

public class Instructions extends AppCompatActivity {  
  
 Button go\_back;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_instructions*);  
  
 getSupportActionBar().hide();  
  
 Window window = this.getWindow();  
 window.addFlags(WindowManager.LayoutParams.*FLAG\_DRAWS\_SYSTEM\_BAR\_BACKGROUNDS*);  
 window.clearFlags(WindowManager.LayoutParams.*FLAG\_TRANSLUCENT\_STATUS*);  
 window.setStatusBarColor(this.getResources().getColor(R.color.*black*));  
  
 go\_back = findViewById(R.id.*go\_back*);  
 go\_back.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent(Instructions.this, MainActivity.class);  
 startActivity(intent);  
 }  
 });  
 }  
}

public class MainActivity extends AppCompatActivity {  
  
 Button instructions, new\_game, new\_group, saved\_group, share;  
 Dialog new\_game\_dialog;  
 DBHelper dbHelper;  
 private int notificationId = 1;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 dbHelper = new DBHelper(this);  
  
 getSupportActionBar().hide();  
  
 Window window = this.getWindow();  
 window.addFlags(WindowManager.LayoutParams.*FLAG\_DRAWS\_SYSTEM\_BAR\_BACKGROUNDS*);  
 window.clearFlags(WindowManager.LayoutParams.*FLAG\_TRANSLUCENT\_STATUS*);  
 window.setStatusBarColor(this.getResources().getColor(R.color.*black*));  
  
 notification();  
  
 instructions = findViewById(R.id.*instructions*);  
 new\_game = findViewById(R.id.*new\_game*);  
 share = findViewById(R.id.*share*);  
  
 instructions.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent(MainActivity.this, Instructions.class);  
 startActivity(intent);  
 }  
 });  
  
 new\_game.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Create\_new\_game\_dialog();  
 }  
 });  
  
 share.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent sendIntent = new Intent();  
 sendIntent.setAction(Intent.*ACTION\_SEND*);  
 sendIntent.putExtra(Intent.*EXTRA\_TEXT*, "Hi! Let's play THE IMPOSTOR together!");  
 sendIntent.setType("text/plain");  
 startActivity(sendIntent);  
 }  
 });  
 }  
  
 public void Create\_new\_game\_dialog(){  
 new\_game\_dialog = new Dialog(this);  
 new\_game\_dialog.setContentView(R.layout.*new\_game\_layout*);  
 new\_game\_dialog.setTitle("NEW GAME");  
 new\_game\_dialog.setCancelable(true);  
 new\_group = new\_game\_dialog.findViewById(R.id.*new\_group*);  
 saved\_group = new\_game\_dialog.findViewById(R.id.*saved\_group*);  
  
 new\_group.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent(MainActivity.this, Group\_setting.class);  
 startActivity(intent);  
 }  
 });  
  
 saved\_group.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent(MainActivity.this, Saved\_group.class);  
 startActivity(intent);  
 }  
 });  
  
 new\_game\_dialog.show();  
 }  
  
 public void notification(){  
 Intent myIntent = new Intent(MainActivity.this , NotifyReceiver.class);  
 myIntent.putExtra("notificationId", notificationId);  
 PendingIntent pendingIntent = PendingIntent.*getBroadcast*(MainActivity.this, 0, myIntent, PendingIntent.*FLAG\_CANCEL\_CURRENT*);  
  
 AlarmManager alarmManager = (AlarmManager)getSystemService(*ALARM\_SERVICE*);  
 Calendar start\_time1 = Calendar.*getInstance*();  
 start\_time1.set(Calendar.*SECOND*, 0);  
 start\_time1.set(Calendar.*MINUTE*, 00);  
 start\_time1.set(Calendar.*HOUR\_OF\_DAY*, 8);  
 long notify\_start\_time1 = start\_time1.getTimeInMillis();  
  
 alarmManager.set(AlarmManager.*RTC\_WAKEUP*, notify\_start\_time1, pendingIntent);  
  
 Calendar start\_time2 = Calendar.*getInstance*();  
 start\_time2.set(Calendar.*SECOND*, 0);  
 start\_time2.set(Calendar.*MINUTE*, 00);  
 start\_time2.set(Calendar.*HOUR\_OF\_DAY*, 14);  
 long notify\_start\_time2 = start\_time2.getTimeInMillis();  
  
 alarmManager.setRepeating(AlarmManager.*RTC\_WAKEUP*, notify\_start\_time2, 1000\*60\*60\*24 , pendingIntent);  
  
 Calendar start\_time3 = Calendar.*getInstance*();  
 start\_time3.set(Calendar.*SECOND*, 0);  
 start\_time3.set(Calendar.*MINUTE*, 00);  
 start\_time3.set(Calendar.*HOUR\_OF\_DAY*, 20);  
 long notify\_start\_time3 = start\_time3.getTimeInMillis();  
  
 alarmManager.setRepeating(AlarmManager.*RTC\_WAKEUP*, notify\_start\_time3, 1000\*60\*60\*24 , pendingIntent);  
 }  
}

public class NotifyReceiver extends BroadcastReceiver {  
 @Override  
 public void onReceive(Context context, Intent intent) {  
  
 int notificationId = intent.getIntExtra("notificationId", 0);  
  
 Intent mainintent = new Intent(context, MainActivity.class);  
 intent.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK* | Intent.*FLAG\_ACTIVITY\_CLEAR\_TASK*);  
 PendingIntent pendingIntent = PendingIntent.*getActivity*(context, 0, mainintent, PendingIntent.*FLAG\_UPDATE\_CURRENT*);  
  
 createNotificationChannel(context);  
 NotificationCompat.Builder notification = new NotificationCompat.Builder(context, "id");  
 notification.setContentTitle("UP FOR A GAME?");  
 notification.setContentText("click to start");  
 notification.setSmallIcon(R.drawable.*impostor*);  
 notification.setWhen(System.*currentTimeMillis*());  
 notification.setAutoCancel(true);  
 notification.setContentIntent(pendingIntent);  
  
 NotificationManager notificationManager = (NotificationManager)context.getSystemService(Context.*NOTIFICATION\_SERVICE*);  
 notificationManager.notify(notificationId, notification.build());  
 }  
  
 private void createNotificationChannel(Context context) {  
 *// Create the NotificationChannel, but only on API 26+ because  
 // the NotificationChannel class is new and not in the support library* if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 CharSequence name = "Notifications";  
 String description = "this is notifications";  
 int importance = NotificationManager.*IMPORTANCE\_DEFAULT*;  
 NotificationChannel channel = new NotificationChannel("id", name, importance);  
 channel.setDescription(description);  
 *// Register the channel with the system; you can't change the importance  
 // or other notification behaviors after this* NotificationManager notificationManager = context.getSystemService(NotificationManager.class);  
 notificationManager.createNotificationChannel(channel);  
 }  
 }  
}

public class Place implements Serializable {  
  
 String place\_name;  
 Boolean checked;  
 Boolean clue;  
  
 public Place(String place\_name) {  
 this.place\_name = place\_name;  
 this.checked = false;  
 this.clue = false;  
 }  
  
 public String getPlace\_name() {  
 return place\_name;  
 }  
  
 public void setPlace\_name(String place\_name) {  
 this.place\_name = place\_name;  
 }  
  
 public Boolean getChecked() {  
 return checked;  
 }  
  
 public void setChecked(Boolean checked) {  
 this.checked = checked;  
 }  
  
 public Boolean getClue() {  
 return clue;  
 }  
  
 public void setClue(Boolean clue) {  
 this.clue = clue;  
 }  
}

public class Player implements Serializable {  
  
 String name;  
 boolean impostor;  
  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) { this.name = name; }  
  
 public boolean isImpostor() {  
 return impostor;  
 }  
  
 public void setImpostor(boolean impostor) {  
 this.impostor = impostor;  
 }  
}

public class Saved\_group extends AppCompatActivity {  
  
 Button go\_back;  
 LinearLayout dynamic;  
 DBHelper dbHelper;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_saved\_group*);  
 dbHelper = new DBHelper(this);  
  
 getSupportActionBar().hide();  
  
 Window window = this.getWindow();  
 window.addFlags(WindowManager.LayoutParams.*FLAG\_DRAWS\_SYSTEM\_BAR\_BACKGROUNDS*);  
 window.clearFlags(WindowManager.LayoutParams.*FLAG\_TRANSLUCENT\_STATUS*);  
 window.setStatusBarColor(this.getResources().getColor(R.color.*black*));  
  
 dynamic = findViewById(R.id.*dynamic*);  
  
 ArrayList<Group> groups = new ArrayList<>();  
 groups = dbHelper.getAllGroups();  
 for(int i=0 ; i<groups.size() ; i++) {  
 Group group = groups.get(i);  
 dynamic.addView(createSavedGroup(group, i));  
 }  
  
 go\_back = findViewById(R.id.*go\_back*);  
 go\_back.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent(Saved\_group.this, MainActivity.class);  
 startActivity(intent);  
 }  
 });  
  
 }  
  
 public RelativeLayout createSavedGroup(Group group, int i) {  
 LayoutInflater vi = (LayoutInflater) getApplicationContext().getSystemService(this.*LAYOUT\_INFLATER\_SERVICE*);  
 RelativeLayout saved\_group\_layout = (RelativeLayout) vi.inflate(R.layout.*saved\_group\_layout*, null);  
 TextView group\_name = saved\_group\_layout.findViewById(R.id.*group\_name*);  
 TextView players\_num = saved\_group\_layout.findViewById(R.id.*players\_num*);  
 TextView impostor\_num = saved\_group\_layout.findViewById(R.id.*impostor\_num*);  
 TextView minutes = saved\_group\_layout.findViewById(R.id.*minutes*);  
 Button edit = saved\_group\_layout.findViewById(R.id.*edit*);  
 Button play = saved\_group\_layout.findViewById(R.id.*play*);  
 Button delete = saved\_group\_layout.findViewById(R.id.*delete*);  
  
 group\_name.setText(group\_name.getText().toString() + group.groupname.toString());  
 players\_num.setText(players\_num.getText().toString() + group.playersnum);  
 impostor\_num.setText(impostor\_num.getText().toString() + group.impostornum);  
 minutes.setText(minutes.getText().toString() + group.minutes);  
  
 edit.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent(Saved\_group.this, Group\_setting.class);  
 intent.putExtra("i", Integer.*toString*(i));  
 startActivity(intent);  
 }  
 });  
  
 play.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 group.resetGroup();  
 Intent intent = new Intent(Saved\_group.this, Game.class);  
 intent.putExtra("group", group);  
 startActivity(intent);  
 *// start game :)* }  
 });  
  
 delete.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 dbHelper.deleteGroup(group);  
 Intent intent = new Intent(Saved\_group.this, Saved\_group.class);  
 startActivity(intent);  
 }  
 });  
  
 RelativeLayout.LayoutParams params = new RelativeLayout.LayoutParams(  
 RelativeLayout.LayoutParams.*WRAP\_CONTENT*, RelativeLayout.LayoutParams.*WRAP\_CONTENT*);  
  
 saved\_group\_layout.setLayoutParams(params);  
  
 return saved\_group\_layout;  
 }  
}