



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
UNIVERSITY OF PIRAEUS



Ατομική Εργασία 1

BrakeTracker Android Application

ΠΑΝΑΓΙΩΤΗΣ ΠΑΠΑΚΩΣΤΑΣ – ΜΠΣΠ2330

Περίγραμμα παρουσίασης

1. Εισαγωγή
2. Σχεδίαση και Υλοποίηση
3. Λειτουργικότητα

Εισαγωγή

- ▶ Η εφαρμογή **BrakeTracker** αποτελεί μια εφαρμογή για κινητές συσκευές (android), η οποία προσφέρει στο χρήστη τη δυνατότητα να καταγράφει απότομα φρεναρίσματα, ενώ βρίσκεται σε ένα όχημα/μεταφορικό μέσο.
- ▶ Ο χρήστης της εφαρμογής έχει στη διάθεση του τις παρακάτω λειτουργίες:
 - Εγγραφή στο σύστημα
 - Καταγραφή απότομων φρεναρισμάτων
 - Προβολή συμβάντων σε λίστα
 - Προβολή συμβάντων σε χάρτη Google
- ▶ Παρακάτω παρουσιάζεται η σχεδίαση της εφαρμογής καθώς και ενδεικτικά screenshots που αφορούν την υλοποίηση και τη λειτουργικότητά της

Σχεδίαση και Υλοποίηση

- ▶ Η εφαρμογή προσφέρει τη δυνατότητα καταγραφής απότομων φρεναρισμάτων, εφόσον ο χρήστης έχει δώσει τα απαραίτητα δικαιώματα
- ▶ Η αποθήκευση των δεδομένων γίνεται σε **τοπική βάση δεδομένων (SQLite)**

Σχεδίαση και Υλοποίηση - Καταγραφή συμβάντων

- ▶ Η καταγραφή των συμβάντων γίνεται με τη χρήση του gps της κινητής συσκευής με βάση συγκεκριμένου κανόνα ($\text{acceleration} < -2 \text{ m/s}^2$)

```
LocationCallback locationCallback = onLocationResult(locationResult) → {
    RL_RLCurrentSpeed.setVisibility(View.VISIBLE);
    if (locationResult == null) {
        return;
    }
    for (Location location : locationResult.getLocations()) {
        float currentSpeed = location.getSpeed(); // speed in m/s
        int roundedSpeed = (int) Math.round(currentSpeed * 3.6); // speed in km/h
        long currentTime = System.currentTimeMillis();

        if (lastTime != 0) {
            float speedDifference = currentSpeed - lastSpeed;
            long timeDifference = currentTime - lastTime; // in milliseconds
            float timeDifferenceSeconds = timeDifference / 1000f;
            acceleration = speedDifference / timeDifferenceSeconds;
            Log.d("Speed: ", String.valueOf(currentSpeed));
            Log.d("Acceleration: ", String.valueOf(acceleration));
            tvCurrentSpeed.setText("Speed: " + roundedSpeed + " km/h");

            // Detect braking when acceleration is below -2 m/s²
            if (acceleration < -2) {
                Log.d("Braking", "msg: Braking detected with acceleration: " + acceleration);
                lon = location.getLongitude();
                Log.d("Lon: ", String.valueOf(lon));
                lat = location.getLatitude();
                Log.d("Lat: ", String.valueOf(lat));
                SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd/MM/yyyy HH:mm");
                Date date = new Date(System.currentTimeMillis());
                timeStamp = simpleDateFormat.format(date);
                Log.d("Time: ", timeStamp);
                Log.d("Acceleration: ", String.valueOf(acceleration));
                long id = database.insert(lon, lat, timeStamp, acceleration);
                if (id == -1) {
                    Log.e("ERROR INSERT TO DB", "id: " + id);
                    Toast.makeText(context, "Failed to save braking point. Please try again.", Toast.LENGTH_SHORT).show();
                    return;
                }
                setNotification(lon, lat, timeStamp);
            }
        }

        lastSpeed = currentSpeed;
        lastTime = currentTime;
    }
};
```

Σχεδίαση και Υλοποίηση - Καταγραφή συμβάντων

- ▶ Η καταγραφή των συμβάντων γίνεται εφόσον ο χρήστης έχει δώσει τα απαραίτητα δικαιώματα

```
/**
 * Handles the result of permission requests.
 */
* @param requestCode The request code passed in requestPermissions()
* @param permissions The requested permissions
* @param grantResults The grant results for the corresponding permissions
*/
@Override 14 usages
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == LOCATION_CODE) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            if (isTracking) {
                startTracking();
            }
        } else {
            Toast.makeText(context: this, text: "Permission denied", Toast.LENGTH_SHORT).show();
        }
    }
}

/**
 * Requests the necessary location permissions from the user.
 */
public void askPermission(){ 2 usages
    if(ActivityCompat.checkSelfPermission(context: this, Manifest.permission.ACCESS_FINE_LOCATION)
        != PackageManager.PERMISSION_GRANTED){
        ActivityCompat.requestPermissions(activity: this, new String[]{ Manifest.permission.ACCESS_FINE_LOCATION }, LOCATION_CODE);
    }
}
```

Σχεδίαση και Υλοποίηση – Βάση Δεδομένων SQLite

- Η αποθήκευση των δεδομένων γίνεται σε τοπική βάση δεδομένων **SQLite**

```
/**
 * Database class for managing the braking point history table.
 * This class is responsible for creating, upgrading, and interacting
 * with the SQLite database.
 */
public class Database extends SQLiteOpenHelper {
    public static final String DATABASE_NAME = "brTrack.db";

    private static final String TABLE_NAME = "history_table";

    public static final String ID = "id";
    public static final String LONGITUDE = "lon";
    public static final String LATITUDE = "lat";
    public static final String TIMESTAMP = "time";
    public static final String ACCELERATION = "acceleration";

    private static Database databaseInstance;

    /**
     * Private constructor to prevent direct instantiation.
     * Use the {@link #getInstance(Context)} method to get the singleton instance.
     *
     * @param context The application context
     */
    public Database(@Nullable Context context) {
        super(context, DATABASE_NAME, null, 2);
    }

    /**
     * Retrieves the singleton instance of the Database class.
     *
     * @param context The application context
     * @return The singleton instance of the Database class
     */
    public static synchronized Database getInstance(Context context) {
        if (databaseInstance == null) {
            databaseInstance = new Database(context.getApplicationContext());
        }
        return databaseInstance;
    }
}
```

```
/**
 * Called when the database is created for the first time.
 * This method will create the history table.
 *
 * @param sqLiteDatabase The database instance
 */
@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) {
    createTable(sqLiteDatabase);
}

/**
 * Called when the database needs to be upgraded. This method will drop the existing
 * table and recreate it.
 *
 * @param sqLiteDatabase The database instance
 * @param oldVersion The old database version
 * @param newVersion The new database version
 */
@Override
public void onUpgrade(SQLiteDatabase sqLiteDatabase, int oldVersion, int newVersion) {
    sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME + ";");
    onCreate(sqLiteDatabase); // Recreate table after dropping it
}

public long insert(double lon, double lat, String timeStamp, float acceleration) {
    ContentValues contentValues = new ContentValues();

    contentValues.put(LONGITUDE, String.valueOf(lon));
    contentValues.put(LATITUDE, String.valueOf(lat));
    contentValues.put(TIMESTAMP, timeStamp);
    contentValues.put(ACCELERATION, String.valueOf(acceleration));

    return getDatabaseInstance().insert(TABLE_NAME, null, contentValues);
}

public Cursor getAll() {
    return getDatabaseInstance().rawQuery("SELECT * FROM " + TABLE_NAME, null);
}

public SQLiteDatabase getDatabaseInstance() {
    return this.getWritableDatabase();
}
```

Σχεδίαση και Υλοποίηση – Ειδοποιήσεις

- ▶ Παρέχονται ειδοποιήσεις στο χρήστη σε κάθε καταγραφή συμβάντος

```
/**
 * Sets a notification to alert the user when braking is detected.
 *
 * @param lon      The longitude of the braking point
 * @param lat      The latitude of the braking point
 * @param timestamp The timestamp of the braking event
 */
@SuppressLint("ScheduleExactAlarm") 1 usage
private void setNotification(Double lon, Double lat, String timestamp){
    Calendar calendar = Calendar.getInstance();
    calendar.add(Calendar.SECOND, 3);

    Intent intent = new Intent( packageContext: this, NotificationReceiver.class);
    intent.putExtra( name: "Lon", lon);
    intent.putExtra( name: "Lat", lat);
    intent.putExtra( name: "Time", timestamp);

    PendingIntent pendingIntent = PendingIntent.getBroadcast( context: this, REQUEST_CODE, intent,
        flags: PendingIntent.FLAG_UPDATE_CURRENT | PendingIntent.FLAG_IMMUTABLE);

    AlarmManager alarmManager = (AlarmManager) getSystemService(ALARM_SERVICE);
    alarmManager.setExact(AlarmManager.RTC_WAKEUP, calendar.getTimeInMillis(), pendingIntent);
}
```

Λειτουργικότητα

- ▶ Έναρξη καταγραφής



Tap
→



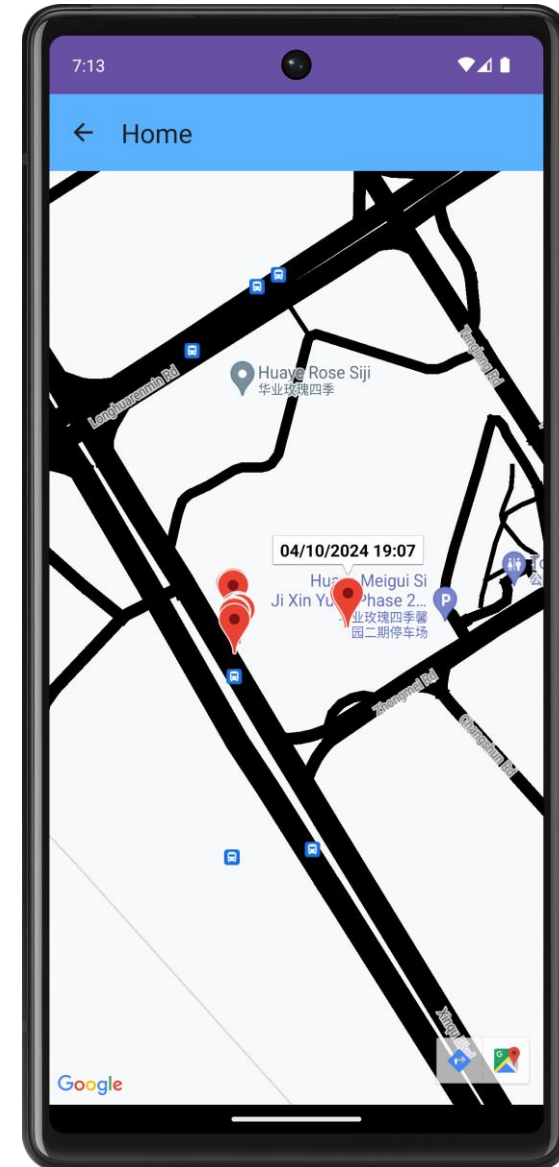
Λειτουργικότητα

- Καταγραφή συμβάντος - Ειδοποίηση



Λειτουργικότητα

- Προβολή συμβάντων σε χάρτη



Λειτουργικότητα

- ▶ Προβολή συμβάντων σε λίστα

