Ατομική Εργασία 2

AutoTrack Android Application

ΠΑΝΑΓΙΩΤΗΣ ΠΑΠΑΚΩΣΤΑΣ – ΜΠΣΠ2330

Περίγραμμα παρουσίασης

1. Εισαγωγή
2. Σχεδίαση και Υλοποίηση
3. Λειτουργικότητα

Εισαγωγή

- Η εφαρμογή **AutoTrack** αποτελεί μια εφαρμογή για κινητές συσκευές (android), η οποία προσφέρει στο χρήστη τη δυνατότητα καταγραφής συμβάντων, ενώ βρίσκεται σε ένα όχημα/μεταφορικό μέσο.
- Ο χρήστης της εφαρμογής έχει στη διάθεση του τις παρακάτω λειτουργίες:
  - Εγγραφή στο σύστημα
  - Καταγραφή συμβάντων:
    - Φρεναρίσματα
    - Απότομες επιταχύνσεις
    - Υπέρβαση ορίου ταχύτητας
    - Έντονες λακκούβες στο οδόστρωμα
  - Προβολή συμβάντων σε λίστα με δυνατότητα εφαρμογής φίλτρων
  - Προβολή συμβάντων σε χάρτη Google Maps με δυνατότητα εφαρμογής φίλτρων
- 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 συμβάντων:
  1. Σε πραγματικό χρόνο όσο ο χρήστης έχει την εφαρμογή σε λειτουργία
  2. Ανά τακτά χρονικά διαστήματα μέσω ενός scheduled job

- Η αποθήκευση των δεδομένων γίνεται σε **απομακρυσμένη βάση δεδομένων (Firebase)**

- Όλοι οι χρήστες της εφαρμογής ταυτοποιούνται μέσω **Firebase authentication**

Σχεδίαση και Υλοποίηση - Καταγραφή συμβάντων

- Η καταγραφή των συμβάντων γίνεται με τη χρήση του gps και των sensors της κινητής συσκευής

```
@Override
public void onCreate() {
    super.onCreate();

    // Initialize location services and sensors
    fusedLocationProviderClient = LocationServices.getFusedLocationProviderClient( context: this
    sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
    accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION);
    database = Database.getInstance();

    // Define location request parameters
    locationRequest = new LocationRequest.Builder( intervalMillis: 5000)
            .setMinUpdateIntervalMillis(2000)
            .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY)
            .build();

    // Location callback to handle location updates
    locationCallback = (LocationCallback) onLocationResult(locationResult) -> {
            if (locationResult == null) return;
            for (Location location : locationResult.getLocations()) {
                handleLocationUpdate(location);
            }
    };

    startForegroundService();
}
```

Σχεδίαση και Υλοποίηση - Καταγραφή συμβάντων

- Έχουν προσδιοριστεί συγκεκριμένοι κανόνες για την καταγραφή του κάθε συμβάντος

```
// Detect rapid acceleration (above 2.5 m/s²)
if (acceleration > 2.5) {
    eventType = EvenyTypeEnum.RAPID_ACCELERATION.getValue();
    Log.d(eventType, msg: "Detected rapid acceleration with acceleration: " + acceleration);
    saveEvent(location, eventType, acceleration);
    setNotification(lon, lat, timeStamp, eventType);
}

// Detect braking when horizontal acceleration is below -2 m/s²
if (acceleration < -2) {
    eventType = EvenyTypeEnum.BRAKING.getValue();
    Log.d(eventType, msg: "Braking detected with acceleration: " + acceleration);
    saveEvent(location, eventType, acceleration);
    setNotification(lon, lat, timeStamp, eventType);
}

// Detect speed limit exceed (speed > 30 km/h)
if (roundedSpeed > 30) {
    eventType = EvenyTypeEnum.SPEED_LIMIT_VIOLATION.getValue();
    Log.d(eventType, msg: "Speed limit exceeded with speed: " + roundedSpeed);
    saveEvent(location, eventType, acceleration);
    setNotification(lon, lat, timeStamp, eventType);
}

// Detect potholes based on vertical acceleration and speed
if (roundedSpeed >= MIN_SPEED_THRESHOLD && Math.abs(lastVerticalAcceleration) > 3.0) {
    // Detecting sudden vertical changes
    eventType = EvenyTypeEnum.POTHOLE.getValue();
    Log.d(eventType, msg: "Pothole detected with vertical acceleration: " + lastVerticalAcceleration);
    saveEvent(location, eventType, lastVerticalAcceleration);
    setNotification(lon, lat, timeStamp, eventType);
}
```

Σχεδίαση και Υλοποίηση - Scheduled Job

- Η έναρξη της καταγραφής γίνεται είτε μέσω της παρέμβασης του χρήστη είτε περιοδικά μέσω scheduled job το οποίο ενεργοποιείται κάθε 30΄ με διάρκεια 5΄

```
private void schedulePeriodicLocationUpdates() {  1 usage
    PeriodicWorkRequest locationRequest = new PeriodicWorkRequest.Builder(LocationWorker.class,  repeatInterval: 30, TimeUnit.MINUTES)
            .setInitialDelay( duration: 0, TimeUnit.SECONDS)
            .build();
    WorkManager.getInstance( context: this).enqueue(locationRequest);

    Toast.makeText( context: this,  text: "Periodic Location Updates Scheduled", Toast.LENGTH_SHORT).show();
}
```

Σχεδίαση και Υλοποίηση - Scheduled Job

```java
public LocationWorker(@NonNull Context context, @NonNull WorkerParameters workerParams) {  no usages
    super(context, workerParams);
}

@NonNull  3 usages
@Override
public Result doWork() {
    Log.d(TAG,  msg: "Periodic Location Update Started");

    // Check permissions before starting the service
    if (ActivityCompat.checkSelfPermission(getApplicationContext(), Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
            ActivityCompat.checkSelfPermission(getApplicationContext(), Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        return Result.failure();
    }

    // Start the TrackingService to collect detailed location and sensor data
    startTrackingService();

    // Schedule the stop of the service after 5 minutes
    scheduler.schedule(this::stopTrackingService, SERVICE_RUNNING_DURATION, TimeUnit.MINUTES);

    return Result.success();
}

private void startTrackingService() {  1 usage
    Context context = getApplicationContext();

    // Create an Intent to start the TrackingService
    Intent trackingServiceIntent = new Intent(context, TrackingService.class);

    // Check for Android Version and start the service accordingly
    if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.O) {
        Log.d(TAG,  msg: "Starting TrackingService as a Foreground Service from WorkManager");
        context.startForegroundService(trackingServiceIntent);
    } else {
        Log.d(TAG,  msg: "Starting TrackingService as a Background Service from WorkManager");
        context.startService(trackingServiceIntent);
    }
}

private void stopTrackingService() {  1 usage
    Context context = getApplicationContext();

    // Create an Intent to stop the TrackingService
```

Σχεδίαση και Υλοποίηση - Λειτουργίες ως service

- Η καταγραφή των συμβάντων πραγματοποιείται μέσω της κλήσης service για βελτιωμένη απόδοση.

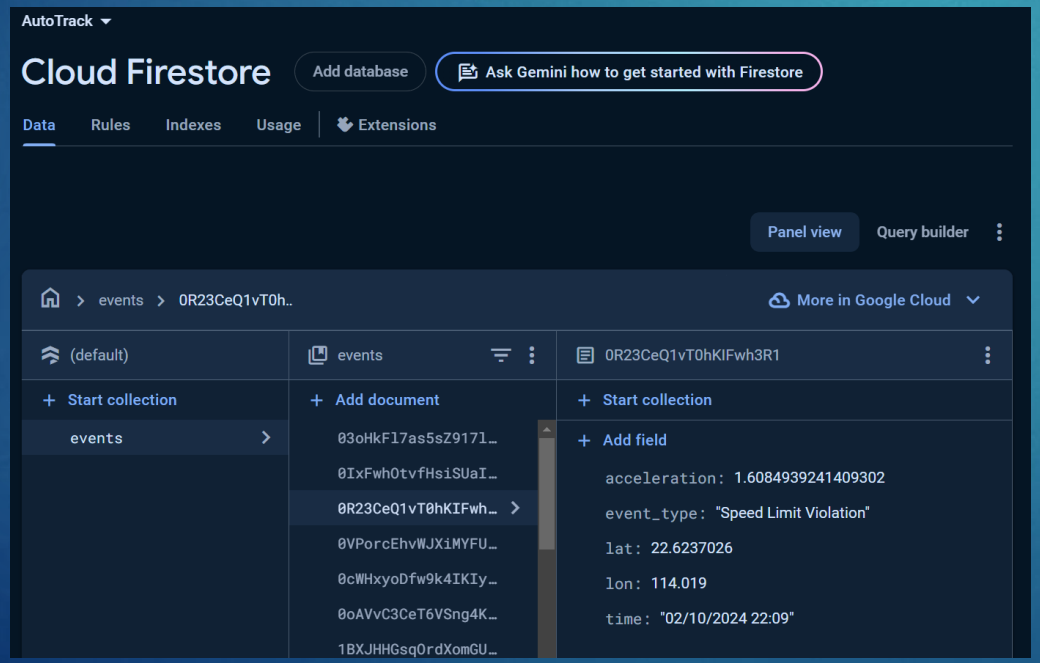
```
// Function to start tracking the user's location
private void startTracking() {  1 usage
    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        askPermission();  // Ask for permission if not already granted
        return;
    }

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        startForegroundService(trackingServiceIntent);
    } else {
        startService(trackingServiceIntent);
    }
    RL_RLCurrentSpeed.setVisibility(TextView.VISIBLE);
    RLStartTracking.startAnimation(pulseAnimation);
    RLStartTracking.setBackground(ContextCompat.getDrawable( context: this, R.drawable.tracking_background));
    Toast.makeText( context: this,  text: "Tracking started", Toast.LENGTH_SHORT).show();
}
```

Σχεδίαση και Υλοποίηση - Λειτουργίες ως service

```
private void startForegroundService() {  1 usage
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        NotificationChannel channel = new NotificationChannel(CHANNEL_ID,  name: "Tracking Service", NotificationManager.IMPORTANCE_LOW);
        NotificationManager manager = getSystemService(NotificationManager.class);
        manager.createNotificationChannel(channel);
    }

    Notification notification = new NotificationCompat.Builder( context: this, CHANNEL_ID)
            .setContentTitle("Tracking Service")
            .setContentText("Tracking your location and movement")
            .setSmallIcon(R.drawable.ic_icon_location)
            .build();

    startForeground( id: 1, notification);
}
```

Σχεδίαση και Υλοποίηση – Βάση Δεδομένων Firebase

- Η αποθήκευση των δεδομένων γίνεται σε απομακρυσμένη βάση δεδομένων **Firebase**




```
public class Database {  13 usages

    private static final String TAG = "Database";  3 usages
    private static final String COLLECTION_NAME = "events";  2 usages

    private FirebaseFirestore db;  3 usages

    // Singleton instance
    private static Database databaseInstance;  3 usages

    private Database() { db = FirebaseFirestore.getInstance(); }

    public static synchronized Database getInstance() {  4 usages
        if (databaseInstance == null) {
            databaseInstance = new Database();
        }
        return databaseInstance;
    }
```

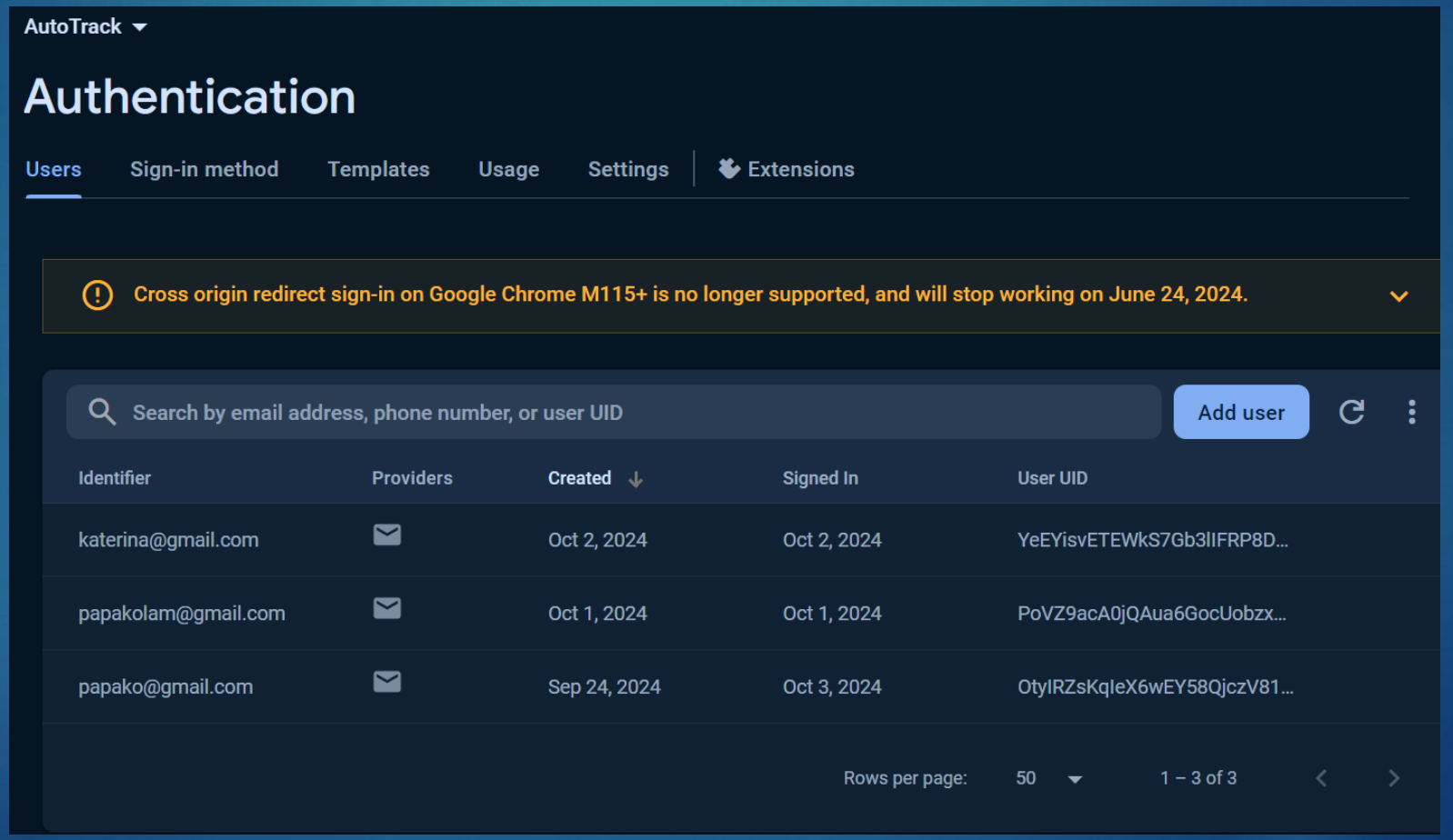
Σχεδίαση και Υλοποίηση – Αυθεντικοποίηση χρηστών

- Οι χρήστες της εφαρμογής ταυτοποιούνται μέσω **Firebase authentication**

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);
    setContentView(R.layout.activity_login);
    FirebaseApp.initializeApp( context: this);

    // Initialize UI elements
    mAuth = FirebaseAuth.getInstance();
    editTextEmail = findViewById(R.id.email);
    editTextPassword = findViewById(R.id.password);
    buttonLogin = findViewById(R.id.btn_login);
    progressBar = findViewById(R.id.progressBar);
    textView = findViewById(R.id.registerNow);

    // Redirect to Register activity
    textView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(getApplicationContext(), Register.class);
            startActivity(intent);
            finish();
        }
    });

    // Login button click listener
    buttonLogin.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) { loginUser(); }
    });
}
```

```java
private void loginUser() { 1 usage
    String email = editTextEmail.getText().toString().trim();
    String password = editTextPassword.getText().toString().trim();
    progressBar.setVisibility(View.VISIBLE);

    if (TextUtils.isEmpty(email)) {
        editTextEmail.setError("Email is required.");
        editTextEmail.requestFocus();
        progressBar.setVisibility(View.GONE);
        return;
    }

    if (TextUtils.isEmpty(password)) {
        editTextPassword.setError("Password is required.");
        editTextPassword.requestFocus();
        progressBar.setVisibility(View.GONE);
        return;
    }

    // Authenticate the user
    mAuth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    progressBar.setVisibility(View.GONE);
                    if (task.isSuccessful()) {
                        Toast.makeText( context: Login.this, text: "Login Successful", Toast.LENGTH_SHORT).show();
                        Intent intent = new Intent(getApplicationContext(), MainActivity.class);
                        startActivity(intent);
                        finish();
                    } else {
                        Toast.makeText( context: Login.this, text: "Authentication failed.", Toast.LENGTH_SHORT).show();
                    }
                }
            });
}
```

Σχεδίαση και Υλοποίηση – Αυθεντικοποίηση χρηστών



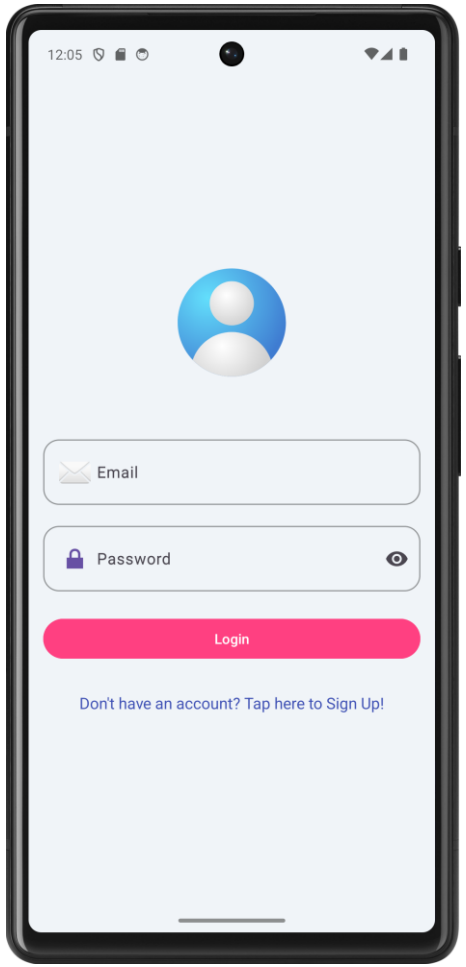
Σχεδίαση και Υλοποίηση – Ειδοποιήσεις

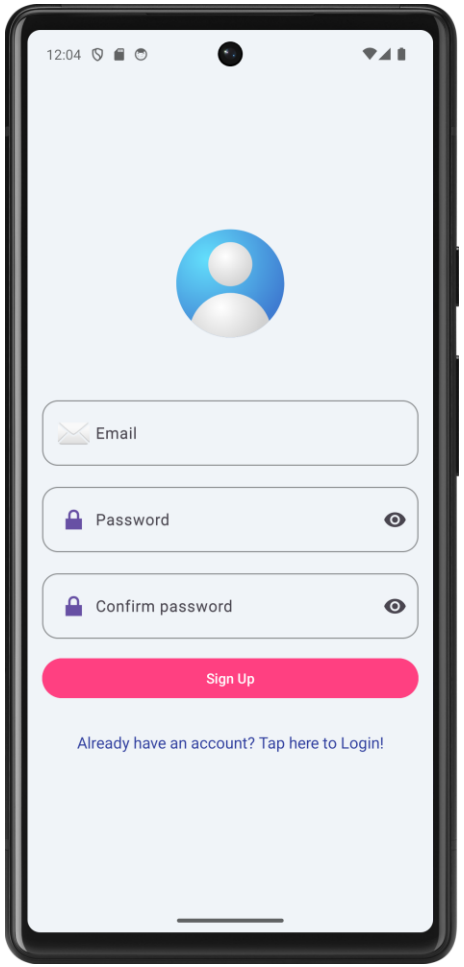
- Παρέχονται ειδοποιήσεις στο χρήστη σε κάθε καταγραφή συμβάντος

```
/**
 * Sets a notification to alert the user when event is detected.
 *
 * @param lon       The longitude of the event
 * @param lat       The latitude of the event
 * @param timestamp The timestamp of the event
 * @param eventType The type of the event
 */
@SuppressLint("ScheduleExactAlarm")  4 usages
private void setNotification(Double lon, Double lat, String timestamp, String eventType){
    Calendar calendar = Calendar.getInstance();
    calendar.add(Calendar.SECOND,  i1: 3);

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

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

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

Λειτουργικότητα

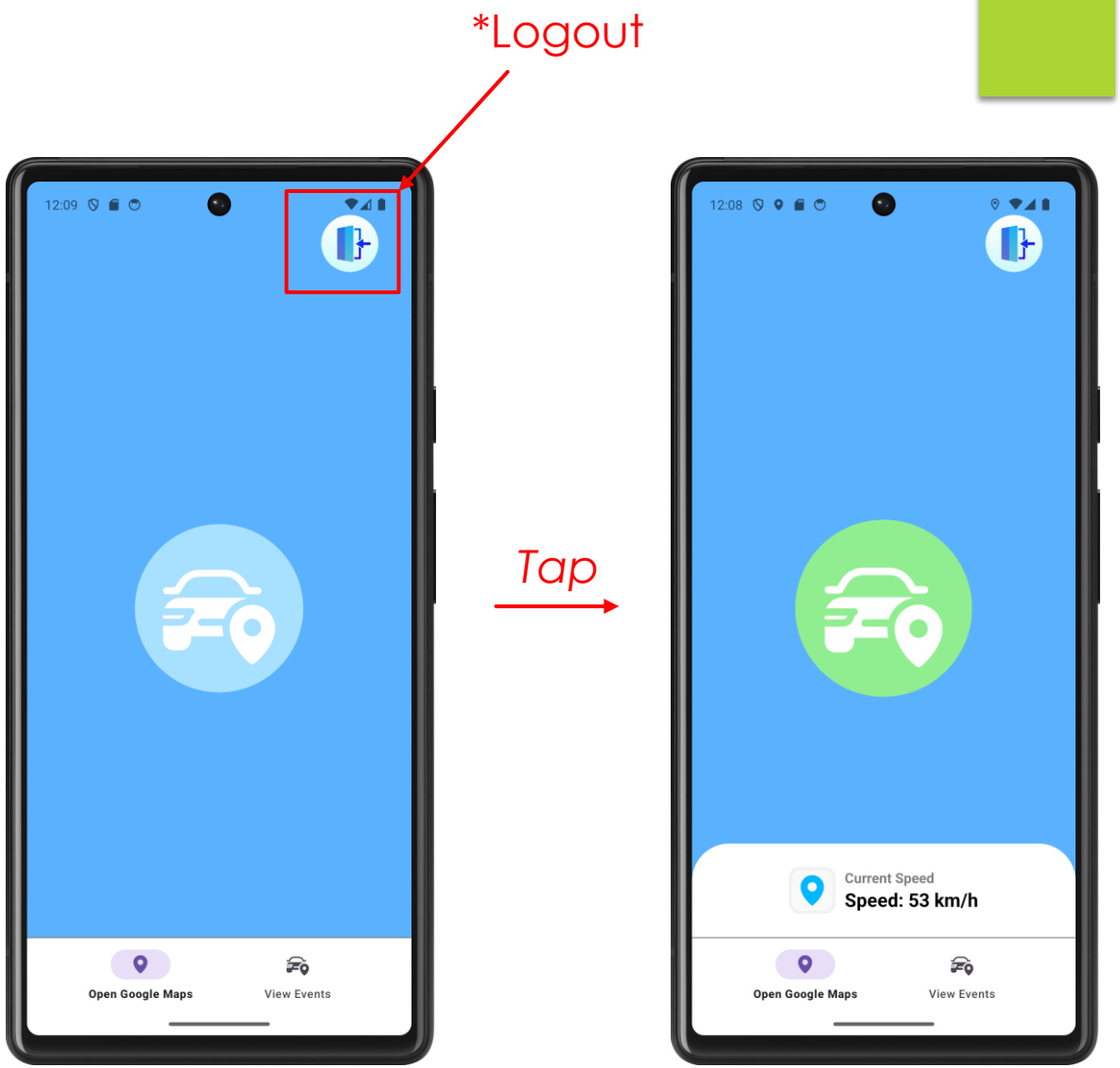
- Login - Register





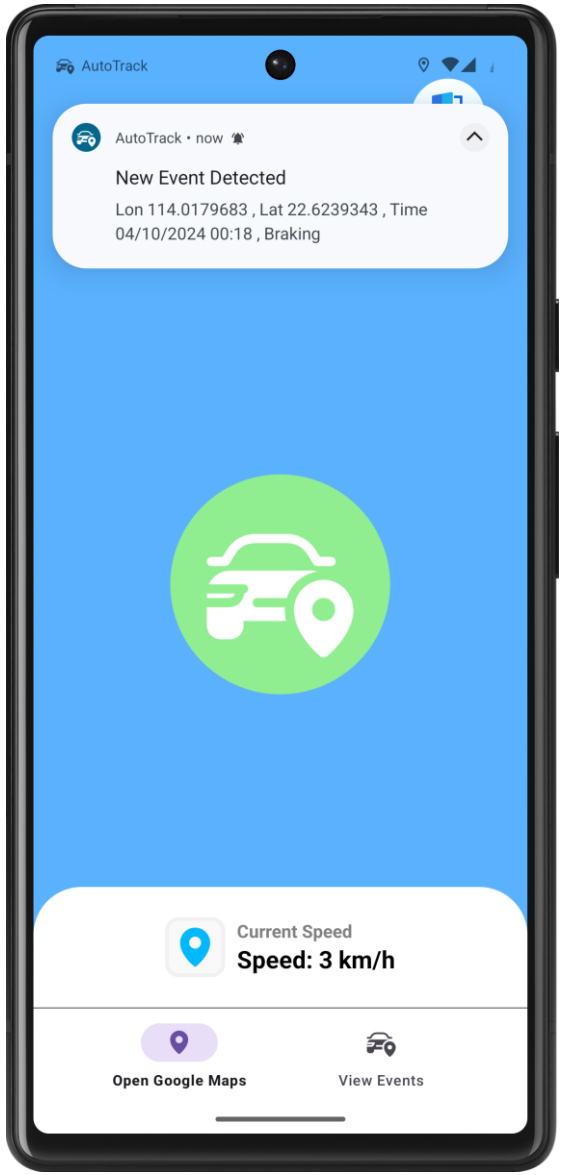
Λειτουργικότητα

- Έναρξη καταγραφής



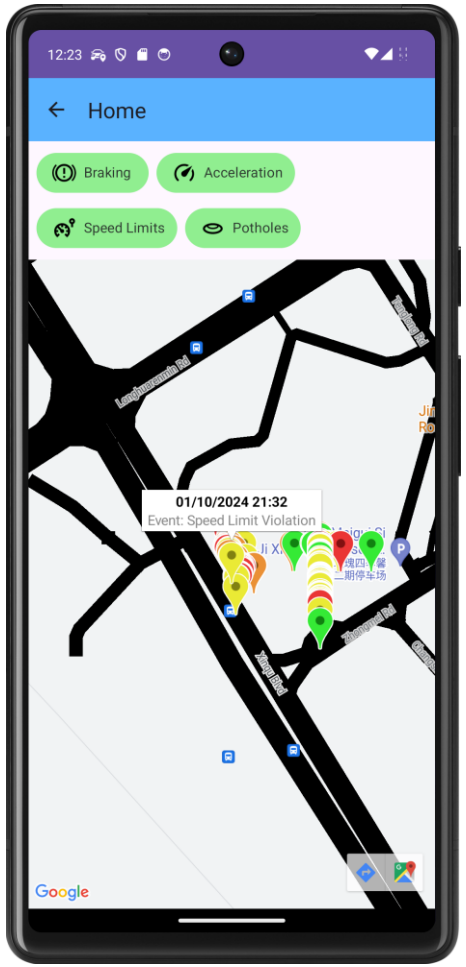
Λειτουργικότητα

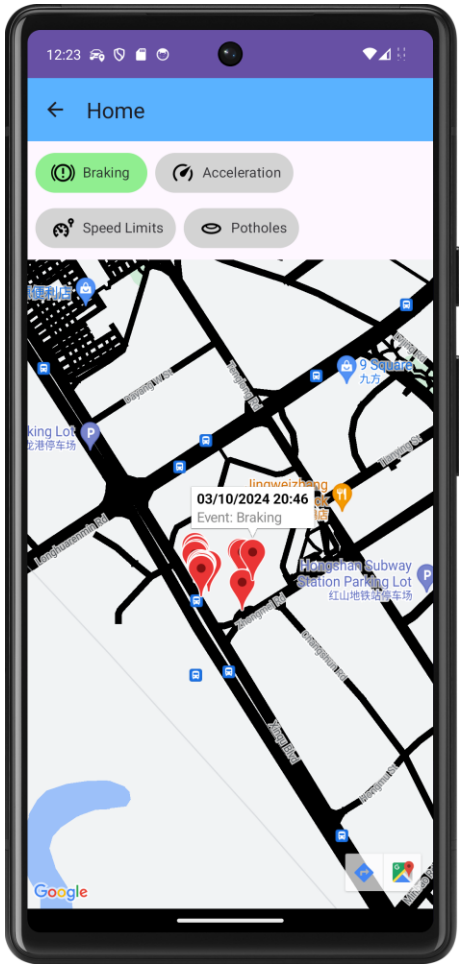
- Καταγραφή συμβάντος - Ειδοποίηση



Λειτουργικότητα

- Προβολή συμβάντων σε χάρτη – εφαρμογή φίλτρων





Λειτουργικότητα

- Προβολή συμβάντων σε λίστα – εφαρμογή φίλτρων