SAFERIDE

Judith Brener, Roy Reinhorn, Eden Barak
THE PROBLEM

• Electric bikes, nowadays, are used vastly, by all ages
• There is limited monitoring (if any)
  • Increasing number of accidents among teenagers
  • Increasing number of death cases among teenagers
• Popularity gains: cheaper bikes and better infrastructures (hopefully)
• Number of casualties rises;
  • 2015- 692
  • 2018- 2185
• Number of deadly accidents rises
  • 2015- 2 deaths
  • 218- 18 deaths (4 under 16)
https://youtu.be/lQ-5qJiAupc
THE PRODUCT - COMPONENTS

Hardware

• The device consists of Adafruit Feather 32u4 Bluefruit LE board.

Battery usage, Activate bike battery

• Accelerometer – based on phone (fall back): identify falls

GPS – based on phone, enables geo-fencing and ride data

Software

• Xamarin based app

• Arduino libraries

• Azure cloud services
HOW DOES IT WORKS - WORKFLOW

- Bluetooth Connection
- HTTP

![Diagram showing components and connections]

- Web App
- Storage Table
- Storage Account
- SQL DB
- Azure
- Notification hub
HARDWARE – THE PARTS

Adafruit Bluefruit 32u4

Bulb
  • Mock on/off of the battery

Potentiometer
  • Mock the voltage inside the battery
Using UART service.
- **UUID**: 6E400001-B5A3-F393-E0A9-E50E24DCCA9E

UART connection
- The service simulates basic UART connection over two lines; TX and RX

The service include two characteristics:
- **TX (0x0002)** - written to by the connected Central device.
- **RX (0x0003)** - used to send data from the peripheral device to the connected Central device.
BLE CONNECTION

Central device scans for Bluetooth low-energy device.
• Blinking Red Led symbols BLE existence.

Connects to our Arduino device
• recognizable by unique prefix name
• Blue Led lights when connection established

Central Device invokes event
• listening to data send from the arduino device (battery percentage change).

Upon start/end ride
• central device sends turn on/off bike instruction to the peripheral device, respectfully.
Each micro-service is:

- Independent - functionality
- Independent - DB
- Independently deployable

Allows:

- Easy maintenance
- Simplicity
- Fast development
NOTIFICATIONS

- To the “Parent” (admin)
- When fall detected
- When exiting the geofence
SERVICES OUTLINE

User management  Smart lock  Buttery usage  Fall detection  Geofance
USERS MANAGEMENT

• Multiple users can use the same device

• “Parent” (Admin) users
  • First Login (no users associated to current device ID) creates new user with admin privileges.

• Login management:
  • Server Data Base
  • Hashed Passwords transmitted (SHA-256)
USERS API

- Manages user’s tables
- Authorization
- Licenses
```
1 { "partitionKey": "123", //DeviceId
2   "rowKey": "Mom", //UserName
3   "isAdmin": true,
4   "password": "bcb9dae6ea88dbf28c262998e6661ec60f32a760faa5aef96745b39c38dbf235" //hashed password (saved)
5 }
```

```
1 { "deviceId": "123",
2   "userName": "Dad",
3   "password": "03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4" //hashed password (confirmation)
5 }
```
<table>
<thead>
<tr>
<th>PARTITIONKEY</th>
<th>ROWKEY</th>
<th>TIMESTAMP</th>
<th>ISADMIN</th>
<th>PASSWORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Dad</td>
<td>2019-06-23T07:22:12.9064153Z</td>
<td>true</td>
<td>03ac674216f3e1f</td>
</tr>
<tr>
<td>123</td>
<td>Son</td>
<td>2019-06-26T15:26:49.0486079Z</td>
<td>false</td>
<td>98aa6675482552</td>
</tr>
<tr>
<td>123</td>
<td>eden</td>
<td>2019-06-26T11:44:42.3513816Z</td>
<td>false</td>
<td>e24ef8f9382887f</td>
</tr>
</tbody>
</table>
SMART LOCK

- Enable engine via application
- Multiple users
- No need for a key
BUTTERY USAGE

• User can see the battery percentage

• //Supply estimated distance according to percentage
GEOFANCE

Safety
- Limits riding area

Hardware
- Device GPS service

Cloud
- SQL DB
- API
- Notification

Algorithm
GEOFANCE API

- StartRide/SaveRiderPlot/FinishRide/SaveRiderGeoFancePlots
- Uses
  - Google maps API
- Logics:
  - Geofence – geometry of space
  - Notifications
  - Rides summary – distance, average speed, etc
**StartRide**

```
{
  "UserId": "1",
  "RiderGeoFanceId": "25",
  "Latitude": 32.106534,
  "Longitude": 34.797006
}
```

**SaveRiderPlot/FinishRide**

```
{
  "RideId": "172",
  "Latitude": -72.2827005,
  "Longitude": 42.9272685,
  "Time": "2019-06-10 19:49:59.010"
}
```

**SaveRiderGeoFancePlots**

```
{
  "UserId": "5",
  "PlotsList": [
    {
      "Longitude": 34.787105, "Latitude": 32.131049
    },
    {
      "Longitude": 34.823497, "Latitude": 32.124362
    },
    {
      "Longitude": 34.815086, "Latitude": 32.107715
    },
    {
      "Longitude": 34.788822, "Latitude": 32.098772
    }
  ]
}
```
# GEOFANCE API - TABLES

**Table 1:**

<table>
<thead>
<tr>
<th>Rideld</th>
<th>Userld</th>
<th>RiderGeoFanceld</th>
<th>StartTime</th>
<th>EndTime</th>
<th>Distance</th>
<th>AverageSpeed</th>
<th>IsInsideGeoFance</th>
</tr>
</thead>
<tbody>
<tr>
<td>140</td>
<td>176</td>
<td>3</td>
<td>2019-06-26 08:27:37.523</td>
<td>2019-06-26 08:28:17.920</td>
<td>0.32</td>
<td>28.67</td>
<td>1</td>
</tr>
<tr>
<td>141</td>
<td>177</td>
<td>3</td>
<td>2019-06-26 08:29:32.700</td>
<td>2019-06-26 08:30:27.920</td>
<td>0.15</td>
<td>10.20</td>
<td>1</td>
</tr>
<tr>
<td>142</td>
<td>178</td>
<td>3</td>
<td>2019-06-26 08:30:57.730</td>
<td>2019-06-26 08:31:32.930</td>
<td>0.16</td>
<td>16.89</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 2:**

<table>
<thead>
<tr>
<th>Userld</th>
<th>Plotsid</th>
<th>Plots.Json</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>[{&quot;Longitude&quot;:42.9284076,&quot;Latitude&quot;:-72.2778296},{&quot;Longitude&quot;:42.9270878,&quot;Latitude&quot;:-72.28698...}</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>[{&quot;Longitude&quot;:42.9284076,&quot;Latitude&quot;:-72.2778296},{&quot;Longitude&quot;:42.9270878,&quot;Latitude&quot;:-72.28698...}</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>[{&quot;Longitude&quot;:42.9284076,&quot;Latitude&quot;:-72.2778296},{&quot;Longitude&quot;:42.9270878,&quot;Latitude&quot;:-72.28698...}</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>[{&quot;Longitude&quot;:42.9284076,&quot;Latitude&quot;:-72.2778296},{&quot;Longitude&quot;:42.9270878,&quot;Latitude&quot;:-72.28698...}</td>
</tr>
</tbody>
</table>

**Table 3:**

<table>
<thead>
<tr>
<th>RidePlotld</th>
<th>Rideld</th>
<th>Longitude</th>
<th>Latitude</th>
<th>Time</th>
<th>IsInsideGeoFance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>42.92810900</td>
<td>-72.27818370</td>
<td>2019-05-25 10:41:53.953</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>42.92749630</td>
<td>-72.28076930</td>
<td>2019-05-24 11:35:45.617</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>42.92726850</td>
<td>-72.28270050</td>
<td>2019-05-22 09:47:59.010</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>42.92810900</td>
<td>-72.27818370</td>
<td>2019-05-25 05:16:28.887</td>
<td>1</td>
</tr>
</tbody>
</table>
GEOFANCE API – ALGORITHM

• Is point inside polygon?

```csharp
public bool CheckInsideGeoFance(int RideId, decimal Latitude, decimal Longitude)
{
    Location point = new Location { Latitude = Latitude, Longitude = Longitude };
    UserRide userRide = riderappEntities.UserRides.Find(RideId);
    RiderGeoFancePlot rp = riderappEntities.RiderGeoFancePlots.Find(userRide.RiderGeoFanceId);
    List<Location> poly = Newtonsoft.Json.JsonConvert.DeserializeObject<List<Location>>(rp.PlotsJson);
    return IsPointInPolygon(poly, point);
}

private static bool IsPointInPolygon(List<Location> poly, Location point)
{
    int i, j;
    bool c = false;
    for (i = 0, j = poly.Count - 1; i < poly.Count; j = i++)
    {
        if (poly[i].Latitude <= point.Latitude && (point.Latitude < poly[j].Latitude) ||
            (poly[j].Latitude <= point.Latitude && (point.Latitude < poly[i].Latitude)) ||
            c = !c;
    }
    return c;
}

private double GetDistanceBetweenLocations(Location location1, Location location2)
{
    double rlat1 = Math.PI * (double)location1.Latitude / 180;
    ```
FALL DETECTION

Safety
• Alert device users

Hardware
• Phone 3-axis accelerometer

Cloud
• SQL DB
• API
• Notification

Algorithm
FALL DETECTION API

- Analyze Accelerometer data
- Calibration
- Handles SQL database
- Users History
FALL DETECTION API - JSONS

```json
1  { 
2    "DeviceId": "123",
3    "UserId": "DAD",
4    "XData": 0.9,
5    "YData": 0.08,
6    "ZData": 0.15,
7    "Time": "2019-06-26T17:03:49.7503776+00:00",
8    "Latitude": 32.012321,
9    "longitude": 34.304213
10 } 
```

```json
1  { 
2    "DeviceID": "1234",
3    "CalibrationX": 0.02,
4    "CalibrationY": 0.1,
5    "CalibrationZ": 0.92
6  } 
```
### FALL DETECTION API - TABLES

#### Calibration Table

<table>
<thead>
<tr>
<th>DeviceID</th>
<th>CalibrationX</th>
<th>CalibrationY</th>
<th>CalibrationZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>0.04</td>
<td>0.13</td>
<td>0.931</td>
</tr>
<tr>
<td>1234</td>
<td>0.02</td>
<td>0.1</td>
<td>0.92</td>
</tr>
</tbody>
</table>

#### Event Log Table

<table>
<thead>
<tr>
<th>fallDetectionID</th>
<th>deviceID</th>
<th>userID</th>
<th>timeStamp</th>
<th>latitude</th>
<th>longitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>479</td>
<td>eden</td>
<td>2019-06-26 14:25:46:207</td>
<td>32.21138400</td>
<td>34.23421000</td>
</tr>
<tr>
<td>2</td>
<td>480</td>
<td>Son</td>
<td>2019-06-25 14:25:46:207</td>
<td>32.34211200</td>
<td>34.31357500</td>
</tr>
<tr>
<td>3</td>
<td>481</td>
<td>eden</td>
<td>2019-06-26 10:20:46:207</td>
<td>32.21134500</td>
<td>34.76563100</td>
</tr>
</tbody>
</table>
FALL DETECTION API - ALGORITHM

- Fall detecting algorithm
THOUGHTS AHEAD

• Gyroscope – more accurate fall detection

• SMS service

• Emergency service

• Multithreaded – allows falls detections and

• Performance