InsuLink

Documentation

Produced by: Nicola Dean and Marco Fasanella

CP: 10674826,10617541

Table of Contents

SECTION 1

Idea

When Type 1 Diabetes[1] is diagnosed, a patient starts a new life with different eyes. From now on, the conception of food is completely different from the normal one, and the patient has to assimilate the big change and learn how to handle the disease. One of the most difficult but at the same time important things that the patient must learn, is the ***carbohydrates count*** and subsequently the correct insulin dose for a bolus [2]. InsuLink has been designed with the main purpose of giving an hand to Type 1 Diabetes patient with the calculation of the correct ***insulin doses*** and storing Glycemia values.

1.1 Main Goal

InsuLink main goal is to give a first support to the patient but only if combined with the doctor supervision. It is important to underline that this application is only defined by an algorithm, and in this kind of diseases ***each patient needs ad hoc treatments***.

```
\usepackage
```

or

```
\usepackage{package}
```

SECTION 2

Functionalities

Insulink offers some useful tools to keep track of the daily routine of a patient.

2.1 Food Scan

It is possible to to scan a given Food BarCode and be redirectd to the FoodDetails page with all necessary data.

2.2 Glycemia

Keep track of your daily Glycemia with intuitive charts and easly with the glycemia insertion tool.

2.3 Insulin Calculator

An algorithm (inside Insulin Calculator class) will retrieve last Glycemia, total amount of carbohydrates, sport activity and all essential data to calculate the optimal insulin dose for the given meal. A more detailed explaintation can be fount in the Insulin Calculator Section.

2.4 Calendar

The user can see a well detailed sight of all previous data, just choosing a date from the InsuLink calendar, that will retrieve all the informations about that day from the database.

SECTION 3

Screens and Navigation

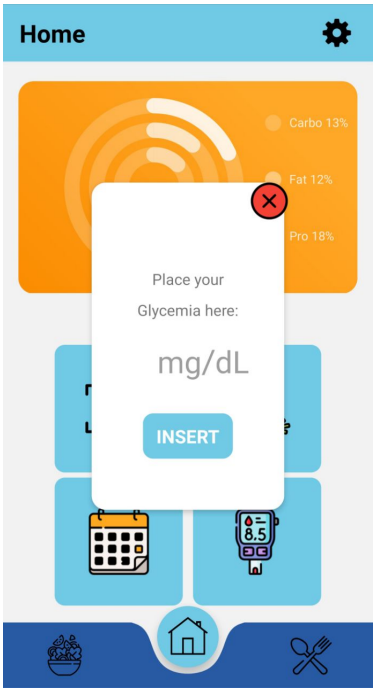
The following provides a screenshot of the pages with a brief description of their use.

3.1 Screens

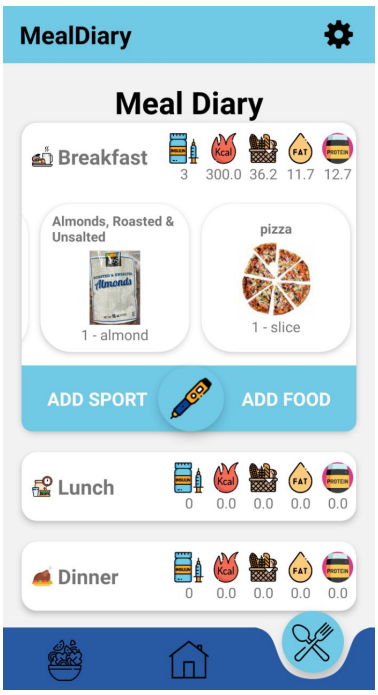
Home Home menu offers shortcuts to the main functionalities and a quick sight of the today glycemia with its intuitive charts.

Search Search food or recipe for nutritional details or to add it in meal diary. User can easly modify the unit measure and quantity of food.

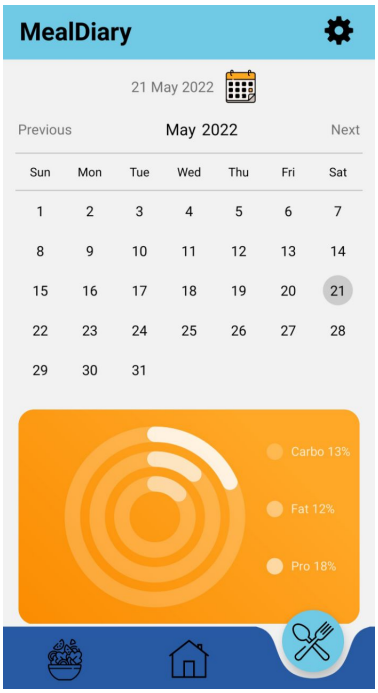
Glycemia PopUp Add glycemia quickly just using the menu shortcut or during the insulin caluclation procedure. The value will automatically stored in Firebase.



Meal Diary Meal Diary can be used for both calulating daily total macro nutrients and insulin dose of each meal.

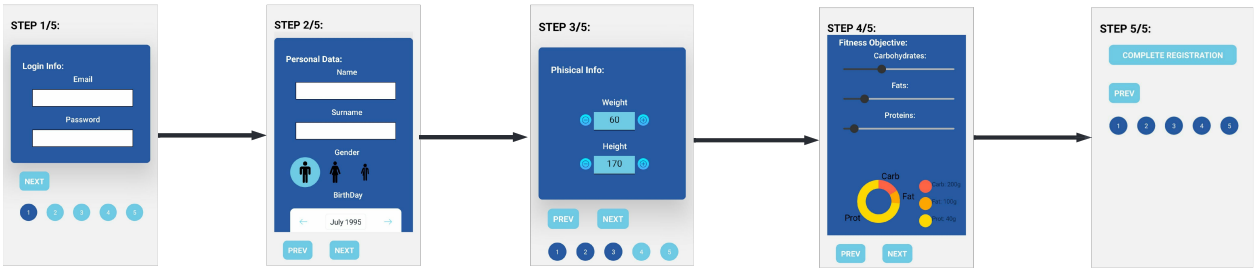


Calendar In calendar it will be possible to retrieve historical data by clicking on a date.

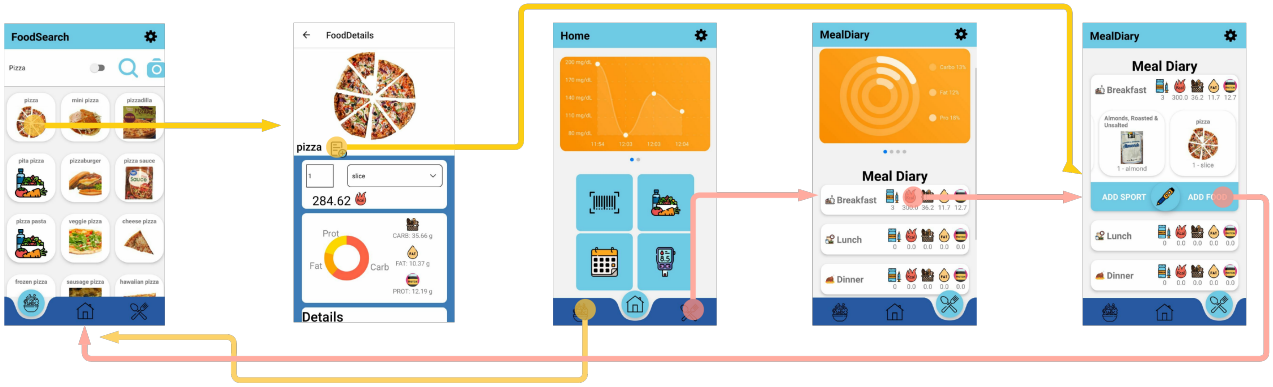


3.2 Navigation

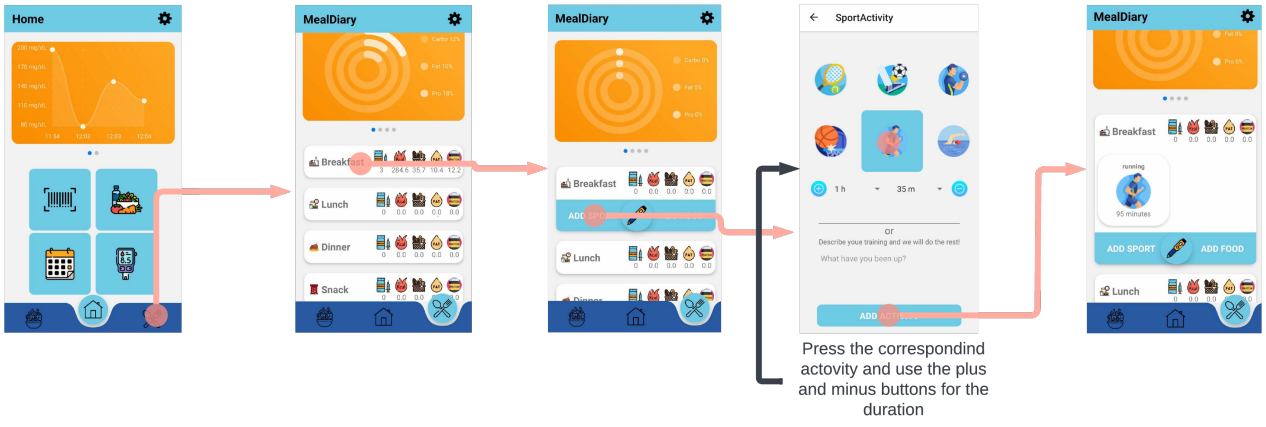
Registration



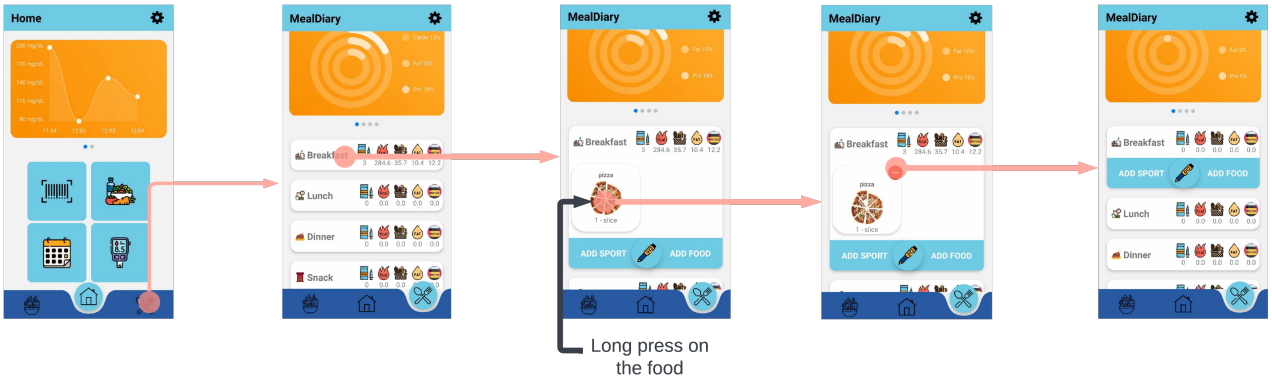
Search and Add Food



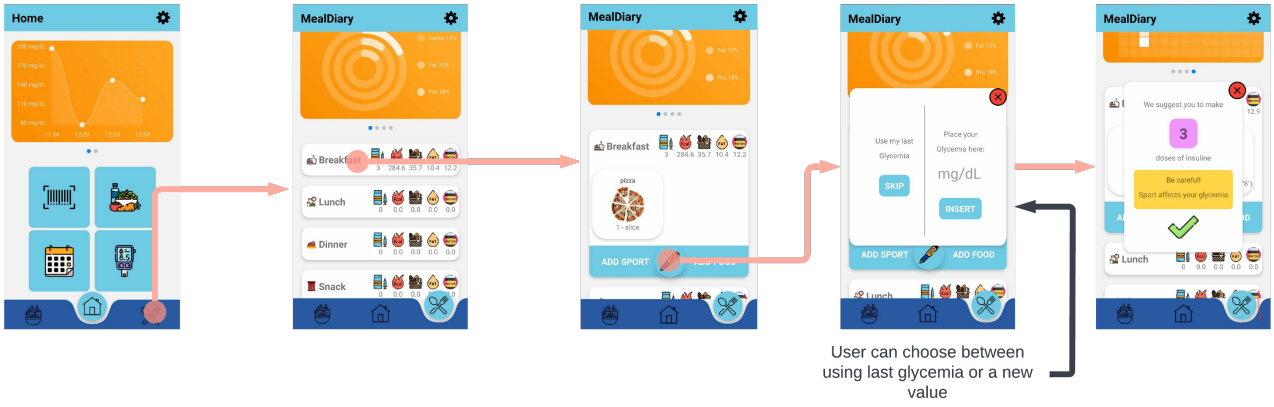
Add Sport



Delete Food



Calculate Insulin

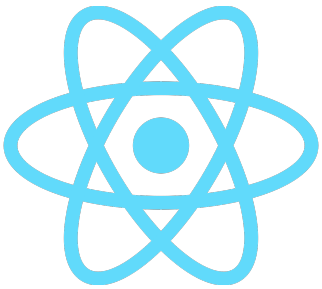


4

SECTION

Architecture

The technology used to make this app is react native [3].



4.1 Folder Structure

`\assets`

Contains all images and component with a proper mapping.

`\constants`

All constants concerning the design and states of the app.

`\customComponents`

All buttons, charts and pickers specifically designed for the pages.

`\pages`

Folder with all pages of the app, using the custom components

`\stateManager`

Redux States for data managing with actions and reducers: macroTracker for meals and userReducer for the patient.

`\utils`

Logic of API, authentication, Firebase and Insulin Calculator

SECTION 5

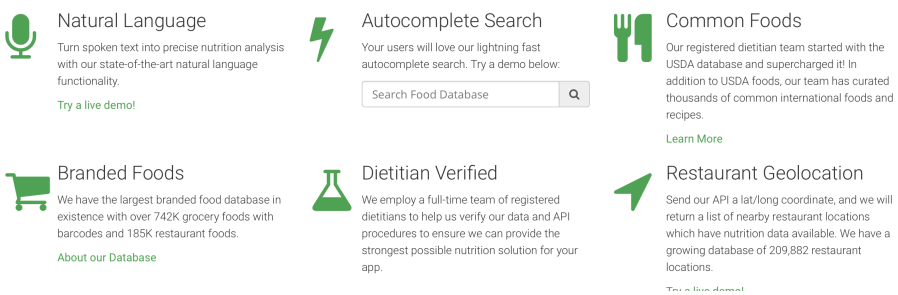
API and Services

InsuLink uses different APIs to get all informations about food and storing user data.

5.1 Nutritionix

Nutritionix [4] is the API used to have a well detailed food database, with all important informations to have a completely working application.

Nutritionix support Natural Language, BarCodes and has a huge dataset with food and recipes. Here there're some functionalities offered by the standard plan:



The API requests and responses are documentend in the ***/utils/apiQuery.js*** class

Methods Using GET requests in **doRequest** method to *trackapi.nutritionix.com/v2*

`\item`

Given upc code (standard use for barcodes) returns the food packaging details.

`\instant`

Given a user query (Food Search page) returns a list of possible foods and recipes.

5.2 Firebase

Firebase [6] is a Google serverless platorm for application development. It is a powerful tool to implement an app database and Google authentication, having many powerful tools to manage it.

All its requests and responses are well documented in the ***src/utils/firebaseQuery.js*** and ***src/utils/auth.js*** classes.

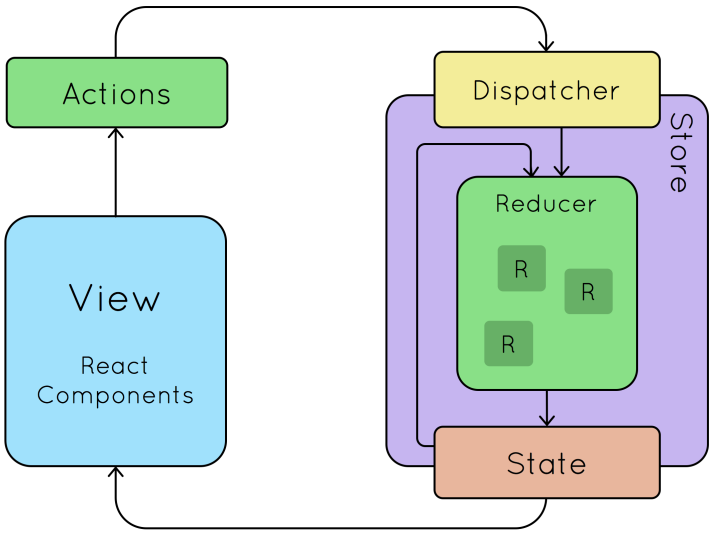
6

SECTION

Redux States

To manage app states we opted for Redux library [8].

Redux is used mostly for application state management. In other words, Redux maintains the state of an entire application in a single immutable state tree (object), which can't be changed directly.
When something changes, a new object is created (using actions and reducers).



The main two reducers of the app are:

`\userReducer`

Contains all data about the user, including some additional fields such as Insulin Sensitivity Factor or CHO Ratio.
If not inserted these values will be calculated considering the patient weight.

```
status:loginStatus.unlogged, //logged,unlogged or pending
userId:"",
mustCompleteReg:false,
userData:{
    email:"",
    password:"",
    name:"",
    surname:"",
    weight:80,
    height:180,
    birthday:{},
    isf:0, //Insulin Sensitivity Factor
    choratio:0, //CHO Ratio
    glicemy:[], //Array of glucose misurations
    maxCarb:200,
    maxFat:100,
    maxProt:40,
}
```

`\macroTracker`

Stores informations about each meal and sport activity of the user. Additionally, cointains the sum of total macros and calories spent during the day.

```
totMacro:{cal:0,carb:0,fat:0,prot:0},
meals:{
    breakfast:{foods:[],macro:{cal:0,carb:0,fat:0,prot:0}},
    lunch:{foods:[],macro:{cal:0,carb:0,fat:0,prot:0}},
    dinner:{foods:[],macro:{cal:0,carb:0,fat:0,prot:0}},
    snack:{foods:[],macro:{cal:0,carb:0,fat:0,prot:0}}
},
history:{
    empty:true,
    totMacro:{cal:0,carb:0,fat:0,prot:0},
    meals:{
        breakfast:{foods:[],macro:{cal:0,carb:0,fat:0,prot:0}},
      //...
    },
    activities:{
        breakfast:{sports:[],totCal:0},
       //...
    }
},
totCalBurned:0,
activities:{
    breakfast:{sports:[],totCal:0},
    //...
}
```

SECTION 7

Insulin Calculator

To understand the algorithm behind the Insulin Calculator, here're some basics:

- Approximately 40-50 percent of the total daily insulin dose is to replace insulin overnight, when you are fasting and between meals
- The other 50-60 percent of the total daily insulin dose is for carbohydrate coverage (food) and high blood sugar correction
- The bolus dose for food coverage is prescribed as an insulin to carbohydrate ratio. The insulin to carbohydrate ratio represents how many grams of carbohydrate are covered or disposed of by 1 unit of insulin.
- The bolus dose for high blood sugar correction is defined as how much one unit of rapid-acting insulin will drop the blood sugar.
- Sport Activity deeply influences the response to insuline and determine a less request of insulin in the patient.

Not considering sport activity, may lead the patient to a low bloodsugar value. Insulink checks if the user has done sport and suggests a new dose value with one unit less. This is not true for all patients, some may need even less amount of insulin.

7.1 Formulas

The main equation for the correct meal dose calculation is:

$$\text{CHO Insulin Dose} = \Sigma \text{ grams of CHO in the meal / grams of CHO disposed by 1 unit of insulin}$$

This happens in ideal conditions where glycemia is in the optimal interval.
Normally this is not the case: after a glucose misuration the patient could find an high bloog sugar. In this case there's a ***correction dose***:

$$\text{High Blood Sugar Correction Dose} = \Delta \text{ (Actual blood sugar - Target blood sugar)} \div \text{Correction Factor}$$

The result formula will be:

$$\text{CHO Total Dose} = \text{CHO Insulin Dose} + \text{High Blood Sugar Correction Dose}$$

Moreover, there's the sport activity to take into account and it will affect the value by decresing it.

8

SECTION

Testing

To perform automated and personalized testing it was used Jest [5] and React Test Renderer [10]. It is a JavaScript Testing Framework that supports React Native. Tests were performed on:

`\__tests__`

Where to overcome some technological barriers, mock objects were used instead of not supported libraries.

8.1 Testing Strategy

The strategy chosen for testing combine different technique such as Snapshots, Render and trigger events, Unit Testing.
Snapshotting have been used to capture the visual aspect of pages or the state of redux reducers, While Rendering and trigger events have been used to test all the navigation/button clicks event and consequences Lastly all the other components, such as firebaseQuery,ApiQuery,InputChecker have been tested using Unit Testing.

8.2 Folders

`\api-test`

Local storage and API calls from Firebase and Nutritionix.

`\redux-test`

Redux and user actions such as: adding food to meal or removing it.

`\renders-test`

Checks correct Pages rendering. Makes snapshots of all pages and compares them with expected result.

`\utils`

Checks User input and Insulin Calculator

At the end were performed more than 50 successful tests on the app.

8.3 Tests

utils	
insuline-calculator	1. Check correct calculation
input-checker	1. Actual Registration Errors 2. Personal Info Errors 3. Physical Info Errors
api-test	
firebase	1. Get User Data 2. Save User Data 3. Get Food Diary 4. Set Food Diary 5. Get Glicemy Data 6. Add Glicemy Data
api-query	1. Search for foods 2. Get Food Details 3. Get Food by barrcode 4. Get Activity calories
local-storage	1. Get User Data 2. Save User Data 3. Get Food Diary 4. Set Food Diary 5. Get Glicemy Data 6. Add Glicemy Data
redux-test	
local-storage	1. Get User Data 2. Save User Data 3. Get Food Diary 4. Set Food Diary 5. Get Glicemy Data 6. Add Glicemy Data
renders-test	

SECTION 9

Future Implementations

Insulink has been structured with the possibility of implementing new technologies inside it.

9.1 API

The API utils section of the code is easly changeble from one provider to another. Using a premium API would affect the performance but also the usability of the app.

9.2 Machine Learning and AI

Calculating the correct insulin dose is a really difficult problem. The factor that influences the output is not only the amount of carbohydrates eaten, but many other features: fats, sport activity, emotional condition and sometimes even the weather.
Moreover each patient needs specific treatments, and has a different resistance to insulin.
Using Machine Learning in this field could be a smart way to correctly predict the optimal insulin dose.

9.3 NFC Glucose Meter

Some Glucose Meters use new technologies to simplify diabetic patients life. One famous example is FreeStyle Libre [7]. Using the NFC technology to retrieve glycemia helps not only in terms of time but also in visualization and store of data.
Once implemented, the user has just to bring the phone closer to the sensor and the app will check the glucose (and store it).

10

SECTION

References

[1] Diabetes Definition
https://en.wikipedia.org/wiki/Type_1_diabetes

[2] Bolus Definition
https://en.wikipedia.org/wiki/Bolus_(medicine)

[3] React Native
https://reactnative.dev

[4] Nutritionix
https://www.nutritionix.com

[5] Jest
https://jestjs.io

[6] Firebase
https://firebase.google.com

[7] FreeStyle Libre
https://www.freestyle.abbott/us-en/home.html

[8] Redux
https://redux.js.org

[9] Source for Insulin Calculator
https://dtc.ucsf.edu/types-of-diabetes/type1/treatment-of-type-1-diabetes/medications-and-therapies/type-1-insulin-therapy/calculating-insulin-dose/

[10] Source for React Test Renderer
https://it.reactjs.org/docs/test-renderer.html