**openCX-*Spearch* Development Report**

Welcome to the documentation pages of Spearch of **openCX**!

You can find here detailed information about the product, hereby mentioned as module, from a high-level vision to low-level implementation decisions, a kind of Software Development Report (see [template](https://github.com/softeng-feup/open-cx/blob/master/docs/templates/Development-Report.md)), organized by discipline (as of RUP):

* Business modeling
  + [Product Vision](https://github.com/softeng-feup/open-cx/blob/master/docs/templates/Development-Report.md#Product-Vision) (DONE)
  + [Elevator Pitch](https://github.com/softeng-feup/open-cx/blob/master/docs/templates/Development-Report.md#Elevator-Pitch) (DONE)
* Requirements
  + [Use Case Diagram](https://github.com/softeng-feup/open-cx/blob/master/docs/templates/Development-Report.md#Use-case-diagram) (DONE)
  + [User stories](https://github.com/softeng-feup/open-cx/blob/master/docs/templates/Development-Report.md#User-stories) (DONE)
  + [Domain model](https://github.com/softeng-feup/open-cx/blob/master/docs/templates/Development-Report.md#Domain-model)
* Architecture and Design
  + [Architectural and design decisions](https://github.com/softeng-feup/open-cx/blob/master/docs/templates)
  + [Technological architecture](https://github.com/softeng-feup/open-cx/blob/master/docs/templates)
  + [Logical architecture](https://github.com/softeng-feup/open-cx/blob/master/docs/templates)
* Implementation
  + [Source code](https://github.com/softeng-feup/open-cx/blob/master/docs/templates)
  + [Issues](https://github.com/softeng-feup/open-cx/blob/master/docs/templates): feature requests, bug fixes, improvements.
* Test
  + [Automated tests](https://github.com/softeng-feup/open-cx/blob/master/docs/templates): Functional tests, integration tests, acceptance tests, as much automated as possible.
* Change management
  + [Issues at Github](https://github.com/softeng-feup/open-cx/blob/master/docs/templates)
* Project management
  + Tasks management tool

So far, contributions are exclusively made by the initial team, but we hope to open them to the community, in all areas and topics: requirements, technologies, development, experimentation, testing, etc.

Please contact us!

Thank you!

*Gustavo Speranzini Tosi Tavares*

*João Miguel dos Santos de Lima Monteiro*

*João Oliveira da Rocha*

*João Paulo Ribeiro Nunes*

*Jorge David Monteiro Pacheco*

**Product Vision**

We aim to provide a solid and reliable solution to lost attendees in conferences, while allowing friends to find each other across the event site.

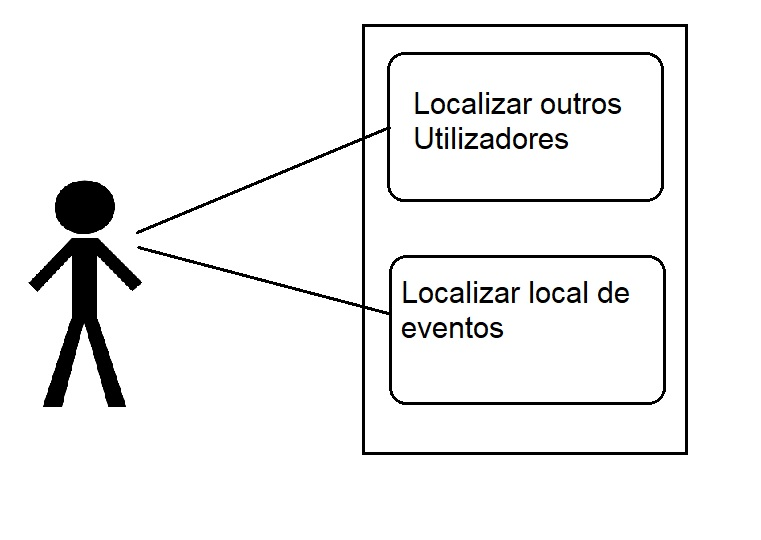
**Elevator Pitch**

Ever got lost at a convention? Having a hard time finding a friend? Just *Spearch* it out! Our product will solve all those your problems.

We aim to deliver a better user experience when attending an event by allowing you to locate your friends or peers, but also finding out where a certain talk will take place, while still being able to maintain your privacy if desired.

**Requirements**

**Use case diagram**

****

* **Actor:** A conference participant who downloaded the app.
* **Description:** Improving the conference’s experience by making it easier for people to find each other or a desired session location.
* **Preconditions**:
  1. Having a smartphone with GPS and Internet connection.
  2. App must be installed.
  3. User must be attending the conference and have an account.
  4. Target users must have also the app and a valid account.
* **Postconditions**.
  1. Describe also the state of the system at the conclusion of the use case execution (postconditions).
* **Normal Flow**.
  1. A user opens the app.
  2. Picks a person or a place, out of the available options;
     1. The app should show a list of the next scheduled events and their location for convenience.
  3. The app should display the path and navigate the person to its destination.
  4. The users can also hide their location.
  5. Users receive a notification when other users start to navigate towards their location.
* **Alternative Flows and Exceptions**.
  1. Target user hides their location mid-navigation: The user gets a notice about the situation and can decide to break navigation or continue to last known location.
  2. User smartphone disconnect from internet or can’t connect to server: User can’t access live information about users. Current navigations try to progress with the offline data.
  3. User smartphone doesn’t have GPS location: User can’t start any navigation and current navigations should halt.

**User stories**

**User story 1:**

I wish I could, as an user, easily find people, like my friends, when attending a conference with them.

**Acceptance Test:**

Given that I have the app, a valid account, GPS, internet connection, Bluetooth and i'm near a beacon (and location visible).

When I search for a friend or other attendee.

Then the app should display their location.

Value: Should Have

Effort: XL

**User story 2:**

As a user, if I am interested in hearing a specific speaker, I would want to find out when and where his next session is.

**Acceptance Test:**

Given that I have the app, a valid account.

When I search for a specific speaker.

Then the app should retrieve information about the next lecture give by that speaker.

Value: Must Have

Effort: M

**User story 3:**

As a participant in a workshop inside the conference, I would like to know where that event is being held.

**Acceptance Test:**

Given that I have the app, internet connection, and GPS / Bluetooth.  
When I'm near a beacon and I choose the event that I want to attend.  
Then the app should display the location and the time of the event.

Value: Must Have

Effort: M

**User story 4:**

As a logged user, I want to choose whether people can track me or not so that I can have some privacy.

**Acceptance Test:**

Given that I have the app, a valid account, GPS / Bluetooth

When I don't want to be disturbed;

Then the app should allow to not share my location.

Value: Could Have

Effort: L

**User Story 5:**

As a user, I want to still have access to events information even without an *active* internet connection, so that I can attend the event and not worry about having a internet connection.

**Acceptance Test:**

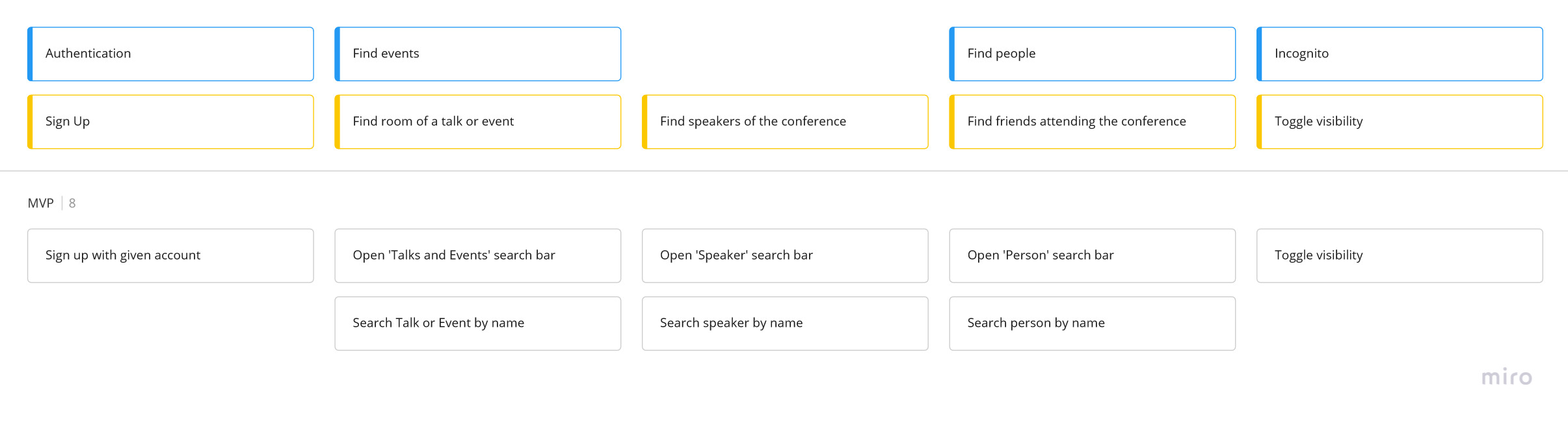
Given that I have the app.

When I lose connection to the internet and still want to search for an event.

Then the app should store information about them and retrieve them.

Value: Could Have

Effort: L



### 

### Domain model

To better understand the context of the software system, it is very useful to have a simple UML class diagram with all the key concepts (names, attributes) and relationships involved of the problem domain addressed by your module.

## Architecture and Design

The architecture of a software system encompasses the set of key decisions about its overall organization.

A well written architecture document is brief but reduces the amount of time it takes new programmers to a project to understand the code to feel able to make modifications and enhancements.

To document the architecture requires describing the decomposition of the system in their parts (high-level components) and the key behaviors and collaborations between them.

In this section you should start by briefly describing the overall components of the project and their interrelations. You should also describe how you solved typical problems you may have encountered, pointing to well-known architectural and design patterns, if applicable.

### Logical architecture

The purpose of this subsection is to document the high-level logical structure of the code, using a UML diagram with logical packages, without the worry of allocating to components, processes or machines.

It can be beneficial to present the system both in a horizontal or vertical decomposition:

* horizontal decomposition may define layers and implementation concepts, such as the user interface, business logic and concepts;
* vertical decomposition can define a hierarchy of subsystems that cover all layers of implementation.

### Physical architecture

The goal of this subsection is to document the high-level physical structure of the software system (machines, connections, software components installed, and their dependencies) using UML deployment diagrams or component diagrams (separate or integrated), showing the physical structure of the system.

It should describe also the technologies considered and justify the selections made. Examples of technologies relevant for openCX are, for example, frameworks for mobile applications (Flutter vs ReactNative vs ...), languages to program with microbit, and communication with things (beacons, sensors, etc.).

### Prototype

To help on validating all the architectural, design and technological decisions made, we usually implement a vertical prototype, a thin vertical slice of the system.

In this subsection please describe in more detail which, and how, user(s) story(ies) were implemented.

## Implementation

Regular product increments are a good practice of product management.

While not necessary, sometimes it might be useful to explain a few aspects of the code that have the greatest potential to confuse software engineers about how it works. Since the code should speak by itself, try to keep this section as short and simple as possible.

Use cross-links to the code repository and only embed real fragments of code when strictly needed, since they tend to become outdated very soon.

## Test

There are several ways of documenting testing activities, and quality assurance in general, being the most common: a strategy, a plan, test case specifications, and test checklists.

In this section it is only expected to include the following:

* test plan describing the list of features to be tested and the testing methods and tools;
* test case specifications to verify the functionalities, using unit tests and acceptance tests.

A good practice is to simplify this, avoiding repetitions, and automating the testing actions as much as possible.

## Configuration and change management

Configuration and change management are key activities to control change to, and maintain the integrity of, a project’s artifacts (code, models, documents).

For the purpose of ESOF, we will use a very simple approach, just to manage feature requests, bug fixes, and improvements, using GitHub issues and following the [GitHub flow](https://guides.github.com/introduction/flow/).