**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)
  + [Elevator Pitch](https://github.com/softeng-feup/open-cx/blob/master/docs/templates/Development-Report.md#Elevator-Pitch)
* Requirements
  + [Use Case Diagram](https://github.com/softeng-feup/open-cx/blob/master/docs/templates/Development-Report.md#Use-case-diagram)
  + [User stories](https://github.com/softeng-feup/open-cx/blob/master/docs/templates/Development-Report.md#User-stories)
  + [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!

*Members:*

*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**

Start by defining a clear and concise vision for your module, to help members of the team, contributors, and users into focusing their often disparate views into a concise, visual, and short textual form. It provides a "high concept" of the product for marketers, developers, and managers.

A product vision describes the essential of the product and sets the direction to where a product is headed, and what the product will deliver in the future.

To learn more about how to write a good product vision, please see also:

* [How To Create A Convincing Product Vision To Guide Your Team, by uxstudioteam.com](https://uxstudioteam.com/ux-blog/product-vision/)
* [Product Management: Product Vision, by ProductPlan](https://www.productplan.com/glossary/product-vision/)
* [Vision, by scrumbook.org](http://scrumbook.org/value-stream/vision.html)
* [How to write a vision, by dummies.com](https://www.dummies.com/business/marketing/branding/how-to-write-vision-and-mission-statements-for-your-brand/)
* [20 Inspiring Vision Statement Examples (2019 Updated), by lifehack.org](https://www.lifehack.org/articles/work/20-sample-vision-statement-for-the-new-startup.html)

**Elevator Pitch**

Draft a small text to help you quickly introduce and describe your product in a short time and a few words, a technique usually known as elevator pitch.

Take a look at the following links to learn some techniques:

* [Crafting an Elevator Pitch](https://www.mindtools.com/pages/article/elevator-pitch.htm)
* [The Best Elevator Pitch Examples, Templates, and Tactics - A Guide to Writing an Unforgettable Elevator Speech, by strategypeak.com](https://strategypeak.com/elevator-pitch-examples/)
* [Top 7 Killer Elevator Pitch Examples, by toggl.com](https://blog.toggl.com/elevator-pitch-examples/)

**Requirements**

In this section, you should describe all kinds of requirements for your module: functional and non-functional requirements.

Start by contextualizing your module, describing the main concepts, terms, roles, scope and boundaries of the application domain addressed by the project.

**Use case diagram**

Create a use-case diagram in UML with all high-level use cases possibly addressed by your module.

Give each use case a concise, results-oriented name. Use cases should reflect the tasks the user needs to be able to accomplish using the system. Include an action verb and a noun.

Briefly describe each use case mentioning the following:

* **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.
* Provide a brief description of the reason for and outcome of this use case, or a high-level description of the sequence of actions and the outcome of executing the use case.
* **Preconditions and Postconditions**.
* Include any activities that must take place, or any conditions that must be true, before the use case can be started (preconditions) and postconditions. Describe also the state of the system at the conclusion of the use case execution (postconditions).
* **Normal Flow**. Provide a detailed description of the user actions and system responses that will take place during execution of the use case under normal, expected conditions. This dialog sequence will ultimately lead to accomplishing the goal stated in the use case name and description. This is best done as a numbered list of actions performed by the actor, alternating with responses provided by the system.
* **Alternative Flows and Exceptions**. Document other, legitimate usage scenarios that can take place within this use case, stating any differences in the sequence of steps that take place. In addition, describe any anticipated error conditions that could occur during execution of the use case, and define how the system is to respond to those conditions.

**User stories**

User story 1:

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

User story 2:

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

User story 3:

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

User story 4:

I want to choose whether people can track me or not so that I can have some privacy.

This section will contain the requirements of the product described as **user stories**, organized in a global **user story map** with **user roles** or **themes**.

For each theme, or role, you may add a small description here. User stories should be detailed in the tool you decided to use for project management (e.g. trello or github projects).

A user story is a description of desired functionality told from the perspective of the user or customer. A starting template for the description of a user story is

*As a < user role >, I want < goal > so that < reason >.*

You add more details after, but the shorter and complete, the better. In order to decide if the user story is good, please follow the INVEST guidelines.

After the user story text, you should add a draft of the corresponding user interfaces, a simple mockups or drafts, if applicable.

For each user story you should write also the acceptance tests (textually in Gherkin), ie, a description of situations that will help to confirm that the system satisfies the requirements addressed in the user story.

At the end, it is good to add a rough indication of the value of the user story to the customers (e.g. MoSCoW method) and the team should add an estimative of the effort to implemente it, in t-shirt sizes (XS, S, M, L, XL).

**Domain model**

A simple UML class diagram with all the key concepts (names, attributes) and relationships involved of the problem domain addressed by your module.