Module « Advanced Cloud »

-

Project « LinkedTek»

Request for proposal

Table of contents

Introduction

The aim of this request of proposal (RFP) is to present the requirements for the LinkedTek project.

1. Company description

EpiLink is a French company which provides a social network for professional relationships. Its main concurrent is the none so famous company LinkedIn.

Founded in 2016 in the dynamic city of Nancy, the society turnover is continually increasing since its creation. The first version of the project is not designed for a such flow of users and several problems occur since few months. The company needs therefore to provide a second version of the project to be compliant with its new constraints.

2. Project context

The provider will have to provide a fully workable professional social network system, the main features will be the abilities to find and add new relations (thanks to some markers like schools, works or even relation of relation “friend of friend”), the abilities to publish some posts and comment them, to see on a single page all the relation’s posts (like a feed on Facebook, Twitter or LinkedIn) and to send messages to the other platform’s users.

Furthermore, the customer would like to have a mobile application of the software in the future, so you, the provider, need to implement a re-usable and “developer-friendly” backend.

It’s not a mandatory but the customer will really enjoy a mobile version of the software, the provider is free to do it or not.

3. Requirements

The software must implement the following requirements and are organized in three categories:

- REQ_DESIGN_XXX: design requirements,
- REQ_FUNC_XXX: functional requirements,
- REQ_IHM_XXX: IHM requirements,
- REQ_DATA_XXX: data managing requirements.

a. Design requirements

REQ_DESIGN_010	You need to release the whole platform (databases included) as a docker.
REQ_DESIGN_020	You must expose a JSON Rest API on the back
REQ_DESIGN_030	You must, at least, respect the Level 2 of the Richardson maturity model
REQ_DESIGN_040	You must provide a functional and well-designed authentication system
REQ_DESIGN_050	You must choose between NodeJS, python, Golang, PHP, java for the backend part
REQ_DESIGN_060	You must provide a JavaScript based frontend (you are free on the JavaScript framework choice)
REQ_DESIGN_070	You must have a persistent storage for all the data (the data needs to stay stored, even after a relaunch of the containers)

REQ_DESIGN_080	You must provide a docker-compose file (docker-compose.yml) with a fully working “docker-compose up” command.
REQ_DESIGN_090	You must be able to manage two kinds of user: administrator and classic user
REQ_DESIGN_100	You must provide a fully tested project, on both part: back and front
REQ_DESIGN_110	You must provide several micro-services; the project needs to be cut under several different micro-services (for scalability and resilience purposes)
REQ_DESIGN_120	Each micro-service needs to be independently scalable of the others (we need to be able to run 5 instances of the user micro-service if we want but only 1 school micro-service for example)
REQ_DESIGN_130	The micro-services communication needs to be language agnostic (JSON / GRPC / …)

b. Functional requirements

REQ_FUNC_010	You must be able to register / log-in on the software
REQ_FUNC_020	You must be able to add / edit / list the schools
REQ_FUNC_030	You must be able to add / edit / list the companies
REQ_FUNC_040	As administrator, you must be able to list all the users
REQ_FUNC_050	As administrator, you must be able to edit / remove / add / ban the users
REQ_FUNC_060	As administrator, you must be able to delete schools or companies
REQ_FUNC_070	As administrator, you must be able to edit / remove / add all the posts and user’s comments.
REQ_FUNC_080	A user (non-administrator) is not able to remove school or companies
REQ_FUNC_090	A user can subscribe or unsubscribe in several companies or in several schools
REQ_FUNC_100	A user can see other users’ profiles (schools, companies, …)
REQ_FUNC_110	A user can add or remove another user to is relations
REQ_FUNC_120	A user has access to a feed with all is relations’ activities (new post, comment, …)
REQ_FUNC_130	A user can add a new post, edit or remove it
REQ_FUNC_140	A user can comment a post (and edit / remove the comment as well)
REQ_FUNC_150	The platform needs to suggest to the user new relation (thanks to common markers like schools, companies or relations)
REQ_FUNC_150	A user can send, see and respond to messages from other users

c. IHM requirements

REQ_IHM_010	You must provide a login / register page (with email (or nickname) / password)
REQ_IHM_020	You must provide a schools / companies management page with these features: - add a school (UNIQUE = check is school not existing before) - remove a school of the connected user (for admin only) - list all the school - Subscribe to a school - Unsubscribe to a school - filter the schools by country, sector - edit a school (for admin only, or edit proposal validate by poll)
REQ_IHM_030	You must provide a profile edition / view: - Schools / works timeline (linked to the school / company entity) - Edit / upload profile picture - Edit / add basic information (age / country / title)

	- View the other users page - Search for other users by name
REQ_IHM_040	You must provide a post management page with these features: - Publish a post - Edit a post (only mine) - Remove a post (only mine) - See other user's post - Comment a post - Edit a comment (only mine) - Remove a comment (only mine)
REQ_IHM_050	You must provide relation management pages with these features: - Add a relation - Remove a relation - Have a relations suggestions page based on schools, companies, …
REQ_IHM_060	You must provide a feed page with these features: - See posts and comment of relations - Being able to comment a post
REQ_IHM_070	The frontend part of the project needs to be user-friendly and nice (even for the administrator panel)

d. Data requirements

REQ_DATA_010	The users and tasks information need to be stored in a persistent database
REQ_DATA_020	A graph database is needed for the relations storage

4. Delivery conditions

The delivery package must contain the following documents:

- Documentary:
  - SAS (Software Architecture Specifications). This document must follow the template "Template – Software Architecture Specifications.docx" given by the company. It must contain an explanation of the provider understanding of the project, constraints and solutions proposed, global and detailed UML diagrams of the architecture proposition, and the state machine diagrams if needed.
  - SQS (Software Qualification Specifications). This document must follow the template 'Template – Software Qualification Specifications.docx" given by the company. It must contain tests procedures to check the software. The document must also contain a traceability matrix which link the requirement id with the test id. The provider should write a first version of this document only from this request for proposal and before any work.
  - SQSA (Software Qualification Specifications Acceptance). This is the SQS document filled with results of the tests procedures.
- Code and binaries:
  - Code. In order to be fully compliant with our existent tools, the software must be packaged and released as docker containers. A docker-compose need to be released too (even if the release only contains one container) as the customer will just need to execute a "docker-compose up" in the project root for starting the project.
  - The customer will always use the last STABLE version of docker and docker-compose
-

- Tests:
    - tests code.

5. Planning

The planning of this project consists of several deadlines, T0 is the beginning of the project:

Object	**Week**
Kick-Off	T0
Follow-Up	T0 + 3 weeks
Follow-Up	T0 + 8 weeks
Follow-Up	T0 + 12 weeks
Delivery of the project	T0 + 12 weeks
Review	T0 + 13 weeks