SOFTWARE REQUIREMENTS SPECIFICATION

for

Experis Academy Case: Lagalt.no

Version 2.2
Winter 2023

By Craig Marais

Noroff Accelerate AS

January 2023

Contents

1 Revision History

Name	Date	Reason For Changes	Version
Initial Version	01.04.2019	Created Document	0.1
Specifications Revision	11.04.2019	Removed FE-08	0.2
Autumn 2019	31.10.2019	Parameterization	1.0
Winter 2020	03.03.2020	Revision	2.0
Winter 2020	03.03.2020	Corrections	2.0.1
Winter 2022	01.03.2022	Updated dates	2.1
Winter 2023	31.01.2023	Date corrections - NAV Fullstack	2.2

2 Introduction

2.1 Purpose

This SRS (Software Requirements Specification) describes the software product to be created by one of the candidate groups in the *Experis Academy* contingent of the *Fullstack Development* short course.

2.2 Document Conventions

For the purposes of this document the following definitions shall apply:

- The *course* refers to the *Fullstack Development* short course as currently offered at *Experis Academy*; specifically the Winter 2023 intake of said course.
- A *candidate* refers to an individual currently enrolled in the course. The plural *candidates* refers to the candidates as they are arranged in groups of four individuals for this case.
- The *software* refers to the final software product to be created.
- *Mentors* refer to the industry experts giving their time to assist the candidates during the case period.

2.3 Intended Audience and Reading Suggestions

This document is intended for use by the mentors and the candidates. Mentors and candidates should read this document and familiarize themselves with the specifications of the software to be created.

2.4 Project Scope

The primary purpose of the software to be a capstone experience for the candidates; they are expected to utilize a selection of possible development options to produce a single software solution that demonstrates their capabilities as developers. The candidates must produce a software solution that is considered a final product. The software is to be produced over a period of 8 weeks.

2.5 Delivery

A deployed version of the software must completed by the morning of the 31^{st} of March 2023. Nearing the completion of the software, candidates will be required to present their solutions twice. A mock presentation will take place on the 29^{th} of March 2023, and the final presentation will take place on the 31^{st} of March.

2.6 References

Candidates should find the following documents on the Noroff Learning Management System[1] (*Moodle*) site:

- Group membership lists with mentors assigned.
- A copy of this document.

[1]`https://lms.noroff.no/`

3 Requirements Specifications

This case describes the specifications and needs for the website to be hosted at `lagalt.no` The site will be used to facilitate connecting individuals in creative fields with projects that need their specific skill sets.

These projects may be in any ‘creative’ field. However, the initial envisaged fields are as follows:

- Music
- Film
- Game Development
- Web Development

Individuals will be able to both create and join projects within these fields.

Before registering a user may browse the projects freely (Anonymous mode), but may not interact directly with them. All search features still work freely in this mode.

Users start by registering an account (using OAuth services), they then have the option to tailor their profile to reflect their own skill sets. These may be displayed publicly or set in a ‘hidden’ mode. Either way once registered a user may apply to join an existing project.

The ‘hidden’ mode is meant for users that want to collaborate but wish to minimise their publicly visible on-line profile.

Alternatively a user may start a project, they must provide a title and short description to start. However, there are many option the user may add in addition, such as linking to GitHub/GitLab pages for the ”in progress” projects. (Other integration options should be considered.) Additional information can always be added/edited at a later stage (from a project’s Admin page).

Once a user has applied to join a project, the project owner(s) can approve/decline the application. During this process they are able to view a users full skill-set (even if the are in ‘hidden’ mode).

If a user is approved they are the able to access all resources associated with the project. This project view should provide convenient means for the collaborators to communicate, such as a persistent chat area & message board.

The website will be entirely in Norwegian (Bokmål).

3.1 Product Perspective

For this case, the candidates are expected to design and develop a distributed software solution, the main components are as follows:

- A web front-end using a modern framework.
- A back-end that exposes the relevant REST API endpoints
- A database (developer's choice)

Consideration must be made for how the system could be scaled to facilitate a large volume of users and traffic.

3.2 Front-end Requirements

The front-end should borrow it's basic design from `reddit.com` (classic view), there must be a listing of current projects as the main view. There should be a 'panel' that allows the user to filter the view.

- **FE-01**: Main view
  This is the main view, and should allow a user to quickly browse current projects.
  The option to login/register should be in-obtrusive.

- **FE-02**: Project at a glance
  The main page should be composed of a collection of project banners that reflect their field/industry and the currently need skills as a measure of the total skills needed.
  The project should also have tags that reflects theme/purpose.
  If the logged in users skill match those needed, then the banner should reflect this visibly.

- **FE-03**: OAuth Registration & Login
  The login and registration should make use of 3rd party OAuth services (IE. Facebook/Google/Twitter/LinkedIn).

- **FE-04**: Search Feature
  Users must have the option to use keyword searches to narrow down the visible projects.

- **FE-05**: Industry switching
  Different industries must be differentiated into different categories.
  It must still be possible to 'view all'.

- **FE-06**: Project View
  Whether logged in or not, it is possible to click on a project to show additional details.
  Public screen/photos/progress should be available to all users, but logged in users should see additional content, such as recent git repos commits as we as the chat panel for the project.
  Logged in users must have an option to apply to join the project, they must acknowledge what information will be shared to the project leader(s) and they must provide a short written motivation.

- **FE-07**: User Profile
  The user must be enter any and all relevant information to their profile.
  The must include a collection of skills, previous accomplished/portfolio, and a place for them to describe themselves.
  The must also be an option to toggle between 'hidden' and regular mode.

- **FE-08**: Project Administration:
  Project leaders must have an administration page that lets them update the project overview, this includes progress updates (public) and marking the over progress (IE. “Founding”, “In progress”, “Stalled”, and “Completed”).
  This is also where they can see project application.

3.3 API Requirements

- **API-01**: The API must have appropriate end points for the data and all endpoints must be implemented in a RESTful manner.
  The API must not use query parameters to pass objects through, that should be done in the Body (application/json).
  However, sorting and filtering **can** make use of query strings/parameters.
- **API-02**: User view history
  The system must maintain a number of lists relating to user activity:
  All project that have been seen from the main page, all projects that have been clicked on, all projects that the users has applied to, and all projects that the user has previously contributed to.
- **API-03**: New content algorithm
  A simple algorithm should be attempted to tailor the new content that a user sees based on previous history/activity.
  This can obviously far exceed the scope of this case, but a simple solution must be attempted.
- **API-04**: Session persistence
  The system must have a means to identify and restore user sessions.
- **API-05**: CORS
  The system must deny requests that are made from unauthorised websites/clients.

3.4 Database Requirements

- **DB-01**: An appropriate database design must be created and documented
  Many parts of this solution will require database tables to store information.
  This is left to the candidates to design.

4 Other Considerations & Requirements

4.1 Security Requirements

4.1.1 SEC-01: User Authentication

Users should be authenticated appropriately before being allowed to interact with the system. All API endpoints (unless otherwise marked) should require an appropriate bearer token to be supplied in the `Authorization` header of the request. 2FA should be enforced[1].

It is recommended to use an external authentication provider (such as Microsoft, Google, or Gitlab) and the *OpenID Connect Auth Code Flow* [2]. If desired, a self-hosted identity provider can be quickly deployed using Docker[3] and Keycloak[4].

Failed authentication attempts should prompt a `401 Unauthorized` response. Authentication related endpoints should also be subject to a rate limiting policy where, if authentication is attempted too many times (unsuccessfully), then requests from the corresponding address should be temporarily ignored. Candidates should decide on an appropriate threshold for rate limiting.

4.1.2 SEC-02: Input Sanitation

All endpoints that accept input from users must ensure that the provided data is appropriately sanitized or escaped so that it cannot be used in an XSS or injection attack.

4.1.3 SEC-03: Credential Storage

Care should be taken to ensure that administrative credentials (i.e. database credentials) are not hard coded into the application and are instead provided to the application using environment variables.

4.1.4 SEC-04: HTTPS

The final, public facing deployment of the application should enforce communication only over HTTPS.

[1] `https://authy.com/what-is-2fa/`
[2] `https://openid.net/specs/openid-connect-core-1_0.html`
[3] `https://hub.docker.com/r/jboss/keycloak/`
[4] `https://www.keycloak.org/`

4.2 Performance Requirements

4.2.1 PRF-01: Loading Time

The system is expected to be responsive at all times and not ‘hang’ or take excessive time to perform actions.

4.2.2 PRF-02: Error Messages

In the event of an error case, the user should be shown a meaningful error message describing what they have done wrong without being confusing. While error messages should be comprehensive, care should be taken not to expose sensitive information that could compromise the security of the application.

4.3 Documentation Requirements

4.3.1 DOC-01: User Manual

Candidates should submit a user manual containing instructions for how to perform each action within the system with accompanying screenshots.

4.3.2 DOC-02: API Documentation

Candidates should submit a detailed listing of each API endpoint they create with the following information:

1. Endpoint method.
2. Endpoint path.
3. Required and accepted headers.
4. Accepted parameters
5. Expected changes to the data.
6. Possible responses and their meanings.
7. Possible error cases with explanations.

4.3.3 DOC-03: README.md

Candidates should provide a `README` file with the following information:

1. A name for the project
2. A list of team members and project participants
3. Detailed installation instructions to run the service from scratch.

The previously mentioned API documentation ***MAY*** also be included in the `README` as opposed to being a separate document.

The user manual ***MAY NOT*** be included as part of the `README` and must be a separate document.

4.4 Other / Miscellaneous Requirements

4.4.1 MISC-01: Deliverables

The candidates are expected to deliver the following artifacts:

1. A `Git` repository containing all code and inline documentation.
2. Any and all project documentation, including plans, database schema and instructional material (such as the user manual).
3. A live, functional deployment of the completed product.
4. A presentation of their product.

4.4.2 MISC-02: Presentation

Candidates must produce a 30 minute presentation which introduces the team, and discuss their product in two parts:
Development (10 min). Candidates must discuss the choices made during development, the difficulties they faced and how they overcame them.
Feature Demonstration (10 min). Candidates must produce a brief overview of the functionality of their final product.
Questions (10 min). Sufficient time should remain for the audience to ask the candidates questions. Candidates should be prepared to defend decisions made throught the development process to the satisfaction of those present.
Candidates should create a 'walk-through' of the core functions of the system. This should include highlighting how the system behaves differently for different user types, and how each user type can achieve their individual goals within the system.