Thought Finder

1. Project description

This application aims to provide an online platform for sharing thoughts with a community. It offers a place where anybody can get involved in open discussions, find solutions to problems shared among the community`s members or simply speaking their mind by sharing ideas and opinions regarding different topics.

2. Technical and functional requirements

Technologies used:

- Backend: ORM + Spring REST
- Frontend: AngularJS

Project-specific requirements:

- The application`s users will have to log in in order to take any action
- Logged-in users will be able to:
  - add/update/view user information (name, password)
  - share/edit/remove thoughts
  - follow thoughts
  - post comments to a thought
  - filter the thoughts feed (all/popular/recent/based on section) or search for thoughts
- There will be a common feed for all users or guests, displaying the thoughts. Guests can view the thoughts but cannot get involved in any way.
- A pie-chart is generated and displays statistics about the most popular sections in the past 30 days.

3. Non-Functional requirements

- **Usability** is the ease of use and learnability of a human-made object such as a tool or device. In software engineering, usability is the degree to which a software can be used by specified consumers to achieve quantified objectives with effectiveness, efficiency, and satisfaction in a quantified context of use.
- **Reliability**
- **Availability** of a system is typically measured as a factor of its reliability – as reliability increases, so does availability. Availability of a system may also be increased by the strategy of focusing on increasing testability, diagnostics and maintainability and not on reliability. Improving maintainability during the early design phase is generally easier than reliability. Maintainability estimates are also generally more accurate. However, because the uncertainties in the reliability estimates are in most cases very large, it is likely to dominate the availability problem, even while maintainability levels are very high. If failures are prevented, none of the others are of any importance and therefore reliability is generally regarded as the most important part.
- **Durability**
- **Adaptability**

4. Use-case scenarios

Use case: **Edit User Info**
Level: User-goal
Primary actor: User
Main success scenario: The user logs in, accesses it`s profile page and can choose to edit profile information. If so, the information is modified in the database.
Extensions: The user logs in, but provides invalid data and an error message is displayed.
The user provides false credentials.

Use case: **Share thought**
Level: User-goal
Primary actor: User
Main success scenario: User tries to log in, if the credentials are correct, the share thought option becomes available and a text section will be opened to add content and sections corresponding to the thought that will be shared with other users.

Use case: **Search for thoughts**
Level: User-goal
Primary actor: User/Guest
Main success scenario: The log-in is no longer required, and anyone can search for a specific thought. On the main page, an input box is revealed, to type the keywords to search for and after the search icon is pressed, the results are showed on the page.

Use case: **User registration and activation**
Level: User-goal
Primary actor: User
Description: A guest can choose to register, in order to be able to share thoughts and post comments. To do so, he has to provide valid: name, username(email) and a password. After the account is created, a mail is sent to the user with an activation key required to activate the account.
Main success scenario: Guest provides valid data, receives the activation key by email and activates the account using the key.
Extensions: Guest provides invalid activation key → Account will not be activated.
Guest provides invalid email → Cannot receive activation key → Cannot activate account.

Use case: **Generate pie chart**
Level: Application-goal
Primary actor: Application services
Description: When a guest/user chooses to view the chart of most popular sections in the past 30 days, a request is sent from the front-end. In the back-end, the Chart Controller receives the request, iterates through the database and searches for valid thought and section information(validates the date interval) and calculates the number of followers for each thoughts, sums it up and provides a total for each section. The data is then sent back to the front-end as a [followers, section] tuple which is interpreted by an external scriptwhich renders the pie chart. Only sections with a number of follows different to 0 are displayed.