Software requirements specification

Introduction

**Purpose**

The goal of this software is to provide a website where companies can post their job offers and students can find these jobs.

This Software Requirements Specification document serves as a guideline for the project and gives a detailed overview of the functionalities of the software. Its aim is to capture the needs of the company RCP and establish a common understanding of the functional and non-functional requirements of the project.

**Stakeholders**

* University students
* External companies
* Owners

**Definition**

User: company

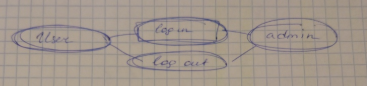
Guest: Student

**System overview**

The system should:

* Show all the accepted job postings on the main page
* Provide an opportunity to CRUD (Create Read Update Delete) operations for job postings
* Be able to open every job postings page with details
* Be able to provide a registration for companies
* Be able to proceed login/logout for companies
* Provide an admin page with additional rights (approve/disapprove new jobs/companies or changes in existing job postings and companies)
* Search and filter the job postings
* Save companies and job postings information in the database

**Use cases**

**\*I’ll do the design later**

1. **Name: Log into the system.**

Description: Company which is already registered visits the website and logs in with his own data.

Preconditions: Company has an account on the website, knows his e-mail address and password

Event Flow:

1. User opens a website and puts the button “Sign in”
2. User fills up e-mail address field
3. User fills up password address field
4. User puts the button “Sign in”.
5. System checks whether the credentials are correct
6. User is led to its homepage. The toaster “Sign In successful” appeared.

Alternate Scenario:

1. User opens a website and puts the button “Sign in”
2. User fills up e-mail address field
3. User fills up password address field
4. User puts the button “Sign in”.
5. System checks whether the credentials are correct
6. In case the e-mail address is correct, the password is incorrect, the toaster “Sign In failed – wrong password” appeared.

Alternate scenario 2:

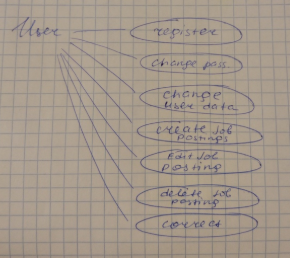
1. User opens a website and puts the button “Sign in”
2. User fills up e-mail address field
3. User fills up password address field
4. User puts the button “Sign in”.
5. System checks whether the credentials are correct
6. In case the e-mail address is incorrect - the toaster “Sign In failed – user not found” appeared.
7. **Name: Log out from the system.**

Description: User which is already registered and signed in logs out from the website.

Preconditions: User has an account on the website, and already logged in

Event Flow:

1. User is on the website
2. User puts the button “Sign out” and is led to the main page. The toaster “You now are signed out” appeared



1. **Name: Registration on the website .**

Description: User registers the website (user fills up all the data about itself) .

Preconditions: User has an internet connection.

Event Flow:

1. User opens a website and puts the button “Register”
2. User fills up all fields (E-mail, password, Confirm password)
3. User choose the user type: company.
4. Company details, Address and Contact Person fields become available
5. User fills up all fields (Company details: Name, Description, Website, Logo; Address: Street/ House No., ZIP code/ place; Contact Person: Name, E-mail, Phone)
6. User puts the button “Register”
7. All the data are sent to database. The toaster “Registration is successful” appeared
8. User is led to the main page (where he can sign in)

Alternate scenario 1:

1. User opens a website and puts the button “Register”
2. User fills up all fields (E-mail, password, Confirm password)
3. User choose the user type: company.
4. Company details, Address and Contact Person fields become available
5. User fills up all fields (Company details: Name, Description, Website, Logo; Address: Street/ House No., ZIP code, Place; Contact Person: Name, E-mail, Phone)
6. User puts the button “Register”
7. In case data in fields “password’ and “confirm password” are not the same the toaster “Registration failed. Invalid Input” appeared. Under the field “confirm password” the message “Password and confirm Password do not match” appeared. User should rewrite information

Alternate scenario 2:

1. User opens a website and puts the button “Register”
2. User fills up fields (E-mail, password, Confirm password)
3. User choose the user type: student
4. The message “This option is currently not available.” appeared

Alternate scenario 3:

1. User opens a website and puts the button “Register”
2. User fills up fields (E-mail, password, Confirm password)
3. User choose the user type: company.
4. Company details, Address and Contact Person fields become available
5. User fills up some of the fields (Company details: Name, Description, Website, Logo; Address: Street/ House No., ZIP code/ place; Contact Person: Name, E-mail, Phone)
6. User puts the button “Register”
7. In case there is no data in any of fields “E-mail”, “Password”, “Confirm Password”, “Name”, “Description”, “Street/ House No.”, “ZIP code”, “Place” the toaster “Registration failed. Invalid Input” appeared. User should fill up information

Postcondition: Data of the new user is saved in the database

1. **Name: Change password**

Description: User wants to change his password

Precondition: User is logged in.

Event Flow:

1. User put the button “My Account”
2. User chooses “Account Settings”
3. User is led to his account page
4. User fills up fields “Password” and “Confirm Password”
5. User puts the button “Save changes”
6. Data is sent to database and toaster “User update successful” appeared.

Alternate Flow:

1. User put the button “My Account”
2. User chooses “Account Settings”
3. User is led to his account page
4. User fills up fields “Password” and “Confirm Password”
5. User puts the button “Save changes”
6. In case data in fields “password’ and “confirm password” are not the same the toaster “User Update failed. Invalid Input” appeared.

Postcondition: New User’s password is saved in the database

1. **Name: Change user data**

Description: User wants to change some data about itself

Precondition: User has an account on the website is logged in.

Event Flow:

1. User put the button “My Account”
2. User chooses “Account Settings”
3. User is led to his account page
4. User changes data in any fields
5. User puts the button “Save changes”
6. Data is sent to database and toaster “User update successful” appeared.

Alternate Flow:

1. User put the button “My Account”
2. User chooses “Account Settings”
3. User is led to his account page
4. User changes data in any fields
5. User puts the button “Save changes”
6. In case there is no data in any of fields “E-mail”, “Password”, “Confirm Password”, “Name”, “Description”, “Street/ House No.”, “ZIP code”, “Place” the toaster “Registration failed. Invalid Input” appeared. User should fill up information.

Postcondition: New User’s data is saved in the database

1. **Name: Create job posting**

Description: User wants to create a job posting

Precondition: User is verified by admin and logged in.

Event Flow:

1. User put the button “My Account”
2. User chooses “My Job Postings”
3. User is led to his Job Postings page (../my-account/job-postings)
4. User puts the button with plus
5. User is led to the page of new job posting creation
6. User fills up all the needed fields.
7. User puts the button “Submit”
8. If all the required fields (fields with sign \*) are fulfilled, then data is sent to admin for verification and toaster “Job creation successful. Now in review of our admin” appeared.

Alternate flow:

1. User put the button “My Account”
2. User chooses “My Job Postings”
3. User is led to his Job Postings page (../my-account/job-postings)
4. User puts the button with plus
5. User is led to the page of new job posting creation
6. User fills up all the needed fields.
7. User puts the button “Submit”
8. If some of the required fields (fields with sign \*) are not fulfilled or written with wrong data type or amount, then the toaster “Job creation failed. Invalid Input” appeared. User should fill up information.

Postcondition: Data about new Job Posting is saved in the database

1. **Name: Edit job posting**

Description: User wants to edit an existing job posting

Precondition: User is verified by admin and logged in, job posting exists.

Event Flow:

1. User put the button “My Account”
2. User chooses “My Job Postings”
3. User is led to his Job Postings page (../my-account/job-postings)
4. User chooses the job posting which he is going to edit and puts the edit button (button with pencil)
5. User is led to the page of job posting edition
6. User fills up/changes all the needed fields.
7. User puts the button “Save changes”
8. Data is sent to admin for verification and toaster “Job updated successfully” appeared.

Alternate flow:

1. User put the button “My Account”
2. User chooses “My Job Postings”
3. User is led to his Job Postings page (../my-account/job-postings)
4. User chooses the job posting which he is going to edit and puts the edit button (button with pencil)
5. User is led to the page of job posting edition
6. User fills up/changes all the needed fields.
7. User puts the button “Save changes”
8. If some of the required fields (fields with sign \*) are not fulfilled or written with wrong data type or amount, then the toaster “Job update failed. Invalid Input” appeared. User should fill up information.

Postcondition: New data about Job Posting is saved in the database

1. **Name: Delete job posting by User**

Description: User wants to delete an existing job posting

Precondition: User is verified by admin and logged in, job posting exists.

Event Flow:

1. User put the button “My Account”
2. User chooses “My Job Postings”
3. User is led to his Job Postings page (../my-account/job-postings)
4. User chooses the job posting which he is going to delete and puts the delete button (button with trash can)
5. The verification message “You are about to delete your job posting. Are you sure?” is appeared.
6. User puts the button “Yes”
7. The job posting is deleted

Postcondition: Data about Job Posting is deleted from the database

1. **Name: Correct**

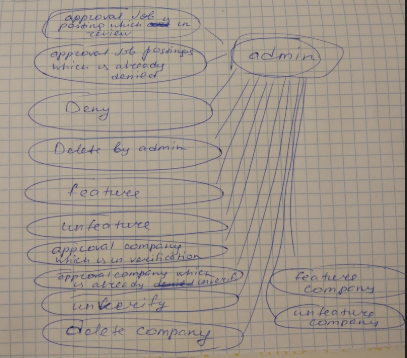
Description: User wants to make all the job postings work

Precondition: User is verified by admin and logged in, user has different job postings

Event Flow:

1. User put the button “My Account”
2. User chooses “My Job Postings”
3. User is led to his Job Postings page (../my-account/job-postings)
4. User puts the group/ungroup button
5. User sees all the job postings, which are sorted for Approves, In Review, and Denied
6. User edit or delete denied job postings

Postcondition: New data about Job Posting is saved or deleted from the database



1. **Name: Approval Job Postings which is In Review**

Description: Admin approves job posting

Precondition: Admin is logged in, job posting exists in review

Event Flow:

1. Admin put the button “Admin”
2. Admin sees the list with two parts “SHOW ALL” and “SHOW IN REVIEW”
3. In “SHOW IN REVIEW” Admin puts “Job Postings”
4. Admin is led to Unaccepted Jobs page (../admin/unaccepted/jobs)
5. Admin reads Jobs information
6. Admin puts the green button
7. The Job Posting disappears from the list. The toaster “Job accepted successfully” appeared. In all Jobs list the mark box changes from “In Review” to “Approved”

Alternate flow:

1. Admin put the button “Admin”
2. Admin sees the list with two parts “SHOW ALL” and “SHOW IN REVIEW”
3. In “SHOW IN REVIEW” Admin puts “Job Postings”
4. Admin is led to Unaccepted Jobs page (../admin/unaccepted/jobs)
5. If there is no Job Posting to verify Admin sees the message “There are currently no open job submissions to approve”

Postcondition: New approval status of the Job Posting is saved in the database

1. **Name: Approval Job Postings which is already denied**

Description: Admin approves job posting

Precondition: Admin is logged in, job posting is already denied

Event Flow:

1. Admin put the button “Admin”
2. Admin sees the list with two parts “SHOW ALL” and “SHOW IN REVIEW”
3. In “SHOW ALL” Admin puts “Job Postings”
4. Admin is led to All Jobs page (../admin/all/jobs)
5. If a Job Posting has red mark box “Denied” admin puts the green button near it
6. The toaster “Job accepted successfully” appeared. The mark box changes from “Denied” to “Approved”

Postcondition: New approval status of the Job Posting is saved in the database

1. **Name: Deny**

Description: Admin denies job posting

Precondition: Admin is logged in, job posting exists

Event Flow:

1. Admin put the button “Admin”
2. Admin sees the list with two parts “SHOW ALL” and “SHOW IN REVIEW”
3. In “SHOW IN REVIEW” Admin puts “Job Postings”
4. Admin is led to Unaccepted Jobs page (../admin/unaccepted/jobs)
5. Admin reads Jobs information
6. Admin puts the red button
7. The message “Reasons why the job submission is denied:” appears
8. Admin fills up the field and puts the button “Ok”
9. The Job Posting disappears from the list. The toaster “Job denied successfully” appeared. In all Jobs list the mark box changes from “In Review” to “Denied”.

Alternate flow 1:

1. Admin put the button “Admin”
2. Admin sees the list with two parts “SHOW ALL” and “SHOW IN REVIEW”
3. In “SHOW IN REVIEW” Admin puts “Job Postings”
4. Admin is led to Unaccepted Jobs page (../admin/unaccepted/jobs)
5. If there is no Job Posting to verify Admin sees the message “There are currently no open job submissions to approve”

Alternate flow 2:

1. Admin put the button “Admin”
2. Admin sees the list with two parts “SHOW ALL” and “SHOW IN REVIEW”
3. In “SHOW ALL” Admin puts “Job Postings”
4. Admin is led to All Jobs page (../admin/all/jobs)
5. Admin puts the red button near the Job Posting which should be denied
6. The message “Reasons why the job submission is denied:” appears
7. Admin fills up the field and puts the button “Ok”
8. The toaster “Job denied successfully” appeared. The mark box changes from “In Review” to “Denied”.

Postcondition: New approval status of the Job Posting is saved in the database

1. **Name: Delete by admin**

Description: Admin deletes job posting

Precondition: Admin is logged in, job posting exists

Event Flow:

1. Admin put the button “Admin”
2. Admin sees the list with two parts “SHOW ALL” and “SHOW IN REVIEW”
3. In “SHOW ALL” Admin puts “Job Postings”
4. Admin is led to All Jobs page (../admin/all/jobs)
5. Admin puts the delete button near the Job Posting which should be deleted
6. The message “Are you sure you want to delete this job item?:” appears
7. Admin puts the button “Ok”
8. The Job Posting disappears from the list. The toaster “Job deleted successfully” appeared.

Postcondition: Data about Job Posting is deleted from the database

1. **Name: Feature**

Description: Admin features job posting

Precondition: Admin is logged in, job posting exists and is not featured

Event Flow:

1. Admin put the button “Admin”
2. Admin sees the list with two parts “SHOW ALL” and “SHOW IN REVIEW”
3. In “SHOW ALL” Admin puts “Job Postings”
4. Admin is led to All Jobs page (../admin/all/jobs)
5. Admin puts the feature button near the Job Posting which should be featured
6. The feature button becomes blue. The toaster “Job Status updated successfully” appeared.

Postcondition: New feature status of the Job Posting is saved in the database

1. **Name: Unfeature**

Description: Admin unfeatures job posting

Precondition: Admin is logged in, job posting exists and is featured

Event Flow:

1. Admin put the button “Admin”
2. Admin sees the list with two parts “SHOW ALL” and “SHOW IN REVIEW”
3. In “SHOW ALL” Admin puts “Job Postings”
4. Admin is led to All Jobs page (../admin/all/jobs)
5. Admin puts the feature button near the Job Posting which should be unfeatured
6. The feature button becomes white. The toaster “Job Status updated successfully” appeared.

Postcondition: New feature status of the Job Posting is saved in the database

1. **Name: Approval Company which is In Verification**

Description: Admin approves company

Precondition: Admin is logged in, company exists in review

Event Flow:

1. Admin put the button “Admin”
2. Admin sees the list with two parts “SHOW ALL” and “SHOW IN REVIEW”
3. In “SHOW IN REVIEW” Admin puts “Companies”
4. Admin is led to Unverified Companies page (../admin/unverified/companies)
5. Admin reads Company information
6. Admin puts the green button
7. The Company disappears from the list. The toaster “Company approved successfully” appeared. In all Companies list the mark box near this company changes from “In Review” to “Verified”

Alternate flow:

1. Admin put the button “Admin”
2. Admin sees the list with two parts “SHOW ALL” and “SHOW IN REVIEW”
3. In “SHOW IN REVIEW” Admin puts “Companies”
4. Admin is led to Unverified Companies page (../admin/unverified/companies)
5. If there is no Company to verify Admin sees the message “There are currently no open company applications to verify.”

Postcondition: New approval status of the Company is saved in the database

1. **Name: Approval Company which is already unverified**

Description: Admin approves company

Precondition: Admin is logged in, company is already unverified

Event Flow:

1. Admin put the button “Admin”
2. Admin sees the list with two parts “SHOW ALL” and “SHOW IN REVIEW”
3. In “SHOW ALL” Admin puts “Companies”
4. Admin is led to All Companies page (../admin/all/companies)
5. If a Company has red mark box “Unverified” admin puts the green button near it
6. The toaster “Company approved successfully” appeared. The mark box changes from “Unverified” to “Verified”

Postcondition: New approval status of the Company is saved in the database

1. **Name: Unverify**

Description: Admin unverifies company

Precondition: Admin is logged in, company exists

Event Flow:

1. Admin put the button “Admin”
2. Admin sees the list with two parts “SHOW ALL” and “SHOW IN REVIEW”
3. In “SHOW IN REVIEW” Admin puts “Companies”
4. Admin is led to Unverified Companies page (../admin/unverified/companies)
5. Admin reads Company information
6. Admin puts the red button
7. The message “Reasons why the user application is denied:” appears
8. Admin fills up the field and puts the button “Ok”
9. The Company disappears from the list. The toaster “Company denied successfully” appeared. In all companies list the mark box changes from “In Verification” to “Denied”.

Alternate flow 1:

1. Admin put the button “Admin”
2. Admin sees the list with two parts “SHOW ALL” and “SHOW IN REVIEW”
3. In “SHOW IN REVIEW” Admin puts “Companies”
4. Admin is led to Unverified Companies page (../admin/unverified/companies)
5. If there is no Company to verify Admin sees the message “There are currently no open company applications to verify.”

Alternate flow 2:

1. Admin put the button “Admin”
2. Admin sees the list with two parts “SHOW ALL” and “SHOW IN REVIEW”
3. In “SHOW ALL” Admin puts “Companies”
4. Admin is led to All Companies page (../admin/all/companies)
5. Admin puts the red button near the Company which should be unverified
6. The message “Reasons why the user application is denied:” appears
7. Admin fills up the field and puts the button “Ok”
8. The toaster “Company denied successfully” appeared. The mark box changes from “In Verification” to “Verified”.

Postcondition: New approval status of the Company is saved in the database

1. **Name: Delete Company**

Description: Admin deletes

Precondition: Admin is logged in, company exists

Event Flow:

1. Admin put the button “Admin”
2. Admin sees the list with two parts “SHOW ALL” and “SHOW IN REVIEW”
3. In “SHOW ALL” Admin puts “Companies”
4. Admin is led to All Companies page (../admin/all/companies)
5. Admin puts the delete button near the Company which should be deleted
6. The message “Are you sure you want to delete this Company?” appears
7. Admin puts the button “Ok”
8. The Company disappears from the list. The toaster “Company deleted successfully” appeared.

Postcondition: Data of the Company is deleted from the database

1. **Name: Feature Company**

Description: Admin features company

Precondition: Admin is logged in, company exists and is not featured

Event Flow:

1. Admin put the button “Admin”
2. Admin sees the list with two parts “SHOW ALL” and “SHOW IN REVIEW”
3. In “SHOW ALL” Admin puts “Companies”
4. Admin is led to All Companies page (../admin/all/companies)
5. Admin puts the feature button near the Company which should be featured
6. The feature button becomes blue. The toaster “Company Status updated successfully” appeared.

Postcondition: New feature status of the Company is saved in the database

1. **Name: Unfeature Company**

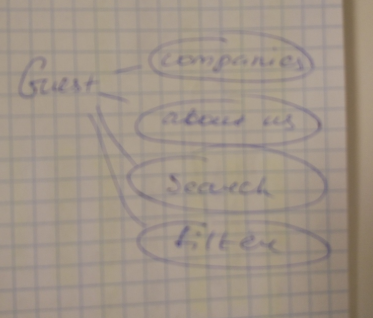
Description: Admin unfeatures company

Precondition: Admin is logged in, company exists and is featured

Event Flow:

1. Admin put the button “Admin”
2. Admin sees the list with two parts “SHOW ALL” and “SHOW IN REVIEW”
3. In “SHOW ALL” Admin puts “Companies”
4. Admin is led to All Companies page (../admin/all/companies)
5. Admin puts the feature button near the Company which should be featured
6. The feature button becomes white. The toaster “Company Status updated successfully” appeared.

Postcondition: New feature status of the Company is saved in the database



1. **Name: Companies**

Description: Guest want to see all the companies

Precondition: Guest has an internet connection

Event Flow:

1. Guest opens the website
2. Guest put the button “Companies” on the header
3. Guest is led to the Companies page
4. **Name: About Us**

Description: Guest want to see information about application

Precondition: Guest has an internet connection

Event Flow:

1. Guest opens the website
2. Guest put the button “About us” on the header
3. Guest is led to the Companies page
4. **Name: Search**

Description: Guest want to search the Job Posting

Precondition: Guest has an internet connection

Event Flow:

1. **Name: Filter**

Description: Guest want to filter the Job Posting

Precondition: Guest has an internet connection

Event Flow:

Actor characteristics:

Guests: the system was developed for any kind of students with undergraduate, graduate, master, etc degree. An approximate age of the target audience is from 20 to 30.

Students use this application for looking for the most appropriate job offers.

Companies: We expect to have both local and worldwide companies.

Companies use the application to allocate the job offers

**Functional and non-functional requirements**

**Functional**

User requirements:

Every user should be able register.

Every user should be able to log-in and log-out of the system.

Every user should have a name, password, and some additional mandatory information. (add exact information which the company should fulfill?)

Every user should be able to add, modify and delete job postings.

Every user should be able to add and modify information about itself.

Guest requirements:

Every guest should be able to search and filter job postings (add in about how?) without registration

Admin requirements:

Admin should be able to log-in and log-out of the system.

Admin should be able to approve, deny, and delete any job postings

Admin should be able to approve or deny new companies and changes in companies data, and delete any company

Surprise feature: Admin should be able to feature and unfeature any company or job postings

Overall requirements:

The application should be able to show all the Job Postings, all the company-partners’ information and the small “about us” page publicly without any registration

We can make more detailed inf about search and filter and other stuff here if we need

**Non-functional**

Security: all the users data should be saved in the database confidently.

Admin should have its own registration check to not let anyone be admin

Platforms: The system should work on all common browsers

The system should work on mobile-platform

User friendly: any system feature should be reachable in 3 clicks