2DV609

Softwork

Requirements Specification

Author1: Jon Cavallie Mester
Author2: Karl Ekberg
Author3: Elias Frigård
Author4: Joel Salo
Author5: Patrik Hasselblad

Table of contents

1. Introduction 4

1.1 Purpose 4

1.2 Scope of the Product 4

1.3 Definitions, acronyms, and abbreviations 4

2. System-wide Requirements 6

2.1 Stakeholders 6

2.2 Requirements 7

2.3 Requirements Analysis 9

2.4 Requirements Classification 11

2.5 Systematic Validation of Functional Requirements 13

2.6 Test Cases for Requirements 14

3. System Interfaces 20

3.1. User Interfaces 20

3.1.1. Look & Feel 20

3.1.2. Layout and Navigation Requirements 21

3.1.3. Consistency 22

3.1.4. User Personalization & Customization Requirements 23

3.2. Interfaces to External Systems or Devices 23

3.2.1. Software Interfaces 23

3.2.2. Hardware Interfaces 24

3.2.3. Communications Interfaces 24

4. Business Rules 24

4.1. Filtering - FG 24

4.1.1. Keyword search FG-1 24

4.1.2. Predefined filters FG-2 24

4.2. User Interface - UI 26

4.2.1. Restrict the number of ads per request UI-1 26

4.2.1. Loading Symbol UI-2 26

4.3. API 26

4.3.1. API unreachable API-1 26

4.4. Ads AD 26

4.4.2. Ad redirection AD-1 26

4.5. Authentication/Authorization AA 27

4.5.1. Anonymous users cannot save ads. AA-1 27

4.5.2. Anonymous users cannot log out. AA-2 27

4.5.3. Save ads. AA-3 27

4.5.4. Unique Users. AA-4 27

4.5.5. Password Length. AA-5 27

5. System Constraints 28

6. Use-Cases 29

6.1. Use-Case: Start application 30

6.3. Use-Case: Login 32

6.6. Use-Case: Remove Favorite 35

6.7. Use-Case: Search 36

6.8. Use-Case: View Ad 38

7. Performance modelling 40

1. Introduction

1.1 Purpose

The purpose of this document is to provide tangible requirements for the software development process of an application that aggregates job ads. The document will provide insight into the attributes of the system, its constraints, and how it should behave. This document should ensure that all stakeholders take part in the process of creating highly qualitative software.

1.2 Scope of the Product

The application ("*Softwork*") will serve as an online web application where users create accounts, browse job ads, and save interesting jobs to check back on later. The main product goal is to help software developers find suitable jobs in their field, and for employers to find employees.

1.3 Definitions, acronyms, and abbreviations

- A **stakeholder** is a person or group that is affected by the software and has a direct, or indirect, influence on the requirements.

- **Functional Requirements (FRs)** describe what the system should do.
    - Example: A user must be able to create an account.

- **Non-functional Requirements (NFRs)** place constraints on how the FRs are implemented.
    - Example: Search results should be presented within 3000 milliseconds.

- **Pre-defined filter** is one or more criteria that are already present within the application and that the user can select to apply to their search.

- **API** (Application Programming Interface), in this context, refers to external APIs from which the application fetches job ads.

1.4 Overview of the remainder of the document

The remainder of this document will lay out the stakeholders, requirements, business rules, system- and interface constraints, test cases, use cases, and performance modelling. Additionally, measures will be taken to ensure the integrity and validity of the requirements:

- Requirements Analysis
- Requirements Classification
- Risk Assessment of the requirements
- Systematic Validation of the requirements

The document is structured in a way where tables, lists, and easy-flowing text will guide the reader and be easy to maintain and update. The following subsections are organized based on the subjects listed in the previous paragraph.

2. System-wide Requirements

2.1 Stakeholders

Internal Stakeholders

- Developers
- Testers
- Product Owners

External Stakeholders

- End-users
- API providers
- External recruiters
- Companies looking to hire

2.2 Requirements

Overview

REQ ID	Functional	Non-Functional	Priority
FR-1	X		Desirable
FR-2	X		Desirable
FR-3	X		Desirable
FR-4	X		Essential
FR-5	X		Desirable
FR-6	X		Useful
FR-7	X		Essential
NFR-1		X	Essential
NFR-2		X	Desirable
NFR-3		X	Useful
NFR-4		X	Essential

The following are functional and non-functional requirements elicited for the specified system. The requirements are identified with a number prefixed by: **FR** = Functional, **NFR** = Non-Functional. These identifiers will be used throughout this document to provide references to requirements.

Functional Requirements (FR)

FR-1 Anonymous users should be able to create an account.

FR-2 Anonymous users should be able to authenticate.

FR-3 Authenticated users should be able to log out.

FR-4 Any user must be able to filter job ads using predefined values.

FR-5 Authenticated users should be able to save a job ad as a favorite.

FR-6 Any user must be able to search job ads by user-defined keywords.

FR-7 Any user must only be presented with jobs related to software development.

Non-functional requirements (NFR)

NFR-1 The application must be able to retrieve existing job ads from external sources.

NFR-2 The application should be usable on both desktop and mobile.

NFR-3 Search results should be presented within an average of 3000 milliseconds.

NFR-4 User credentials must be stored safely.

2.3 Requirements Analysis

Question	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	NFR-1	NFR-2	NFR-3	NFR-4
Premature design	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
Combined requirements	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
Unnecessary requirement	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
Use of non-standard hardware	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
Conformance with business goals	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
Requirements ambiguity	NO	NO	NO	NO	NO	NO	NO	**YES**	NO	NO	**YES**
Requirements realism	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
Requirements testability	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	**NO**

Problems

NFR-1

It can be hard to be more specific as to what sourcing job ads from external sources means. For example, it can mean fetching job ads from an external API or scraping their website for information.

NFR-4

NFR-4 could be interpreted differently depending on the context of the reader and the technical solution later decided for how the authentication process should be implemented.

The testability of NFR-4 relies on the specific implementation that is chosen to solve the problem. Any implementation decisions have not yet been as that would mean the requirement stated is premature. The testability of this requirement will be dependent on the implementation details and security testing could be performed by professional penetration testers to analyse the security of the application.

2.4 Requirements Classification

Keyword	System	User Interface	Database	Communications	Security	Search
FR-1		X	X		X	
FR-2		X	X		X	
FR-3		X			X	
FR-4		X		X		X
FR-5		X	X			
FR-6		X		X		X
FR-7	X	X				X
NFR-1	X			X		
NFR-2	X	X				
NFR-3	X	X		X		
NFR-4			X		X	

System: Requirements that affect the entire system, ex. performance or reliability.

User Interface: Requirements that are related to user interaction.

Database: Requirements that affect data managed by the system in one way or another.

Communications: Requirements that are related to external communication in the system, such as third-party providers.

Security: Requirements that affect the security of either the user and/or application.

Search: Requirements related to searching for specific data from a dataset.

2.5 Systematic Validation of Functional Requirements

Requirement Number	**FR-1**	**FR-2**	**FR-3**	**FR-4**	**FR-5**	**FR-6**	**FR-7**	**NFR-1**	**NFR-2**	**NFR-3**	**NFR-4**
Are the requirements complete?	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
Are the requirements consistent?	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
Are the requirements comprehensible?	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
Are the requirements ambiguous?	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
Is the requirement document structured?	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
Are the requirements traceable?	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
Do the requirements conform to its standards?	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES

2.6 Test Cases for Requirements

Identifier	**FR-1**
Related requirements	FR-2, FR-3
Description	Account creation
Requirement type	Functional
Test steps	1. User provides the required information 2. User signs up.
Expected result	An account has been created.

Identifier	**FR-2**
Related requirements	FR-1, FR-3
Description	User signs in
Requirement type	Functional
Test steps	1. User provides the required information 2. User signs in.
Expected result	The user has been authenticated.

Identifier	**FR-3**
Related requirements	FR-1, FR-2
Description	User signs out
Requirement type	Functional
Test steps	1. User chooses to sign out.
Expected result	The user has been signed out.

Identifier	**FR-4**
Related requirements	NFR-1
Description	Filter jobs using predefined values
Requirement type	Functional
Test steps	1. User selects some predefined value(s) 2. User submits the search.
Expected result	The user gets presented with jobs conforming to the filter criteria.

Identifier	**FR-5**
Related requirements	NFR-1
Description	User saves ads
Requirement type	Functional
Test steps	1. User saves an ad
Expected result	The saved ad is stored.

Identifier	**FR-6**
Related requirements	NFR-1
Description	Search jobs by keywords
Requirement type	Functional
Test steps	1. Search for ads containing a keyword.
Expected result	Job ads containing the keyword are presented.

Identifier	**FR-7**
Related requirements	NFR-1
Description	Only software development jobs
Requirement type	Functional
Test steps	1. Search for a job
Expected result	Only software development jobs matching the keyword are presented.

Identifier	**NFR-1**
Related requirements	None
Description	Jobs sourced externally
Requirement type	Non-Functional
Test steps	1. User starts the application. 2. User submits a search.
Expected result	The application fetches job ads from external sources and presents them to the user.

Identifier	**NFR-2**
Related requirements	None
Description	Application usable on both desktop and mobile
Requirement type	Non-Functional
Test steps	1. Navigate to application on mobile 2. Navigate to application on desktop
Expected result	The application looks good on both mobile and desktop.

Identifier	**NFR-3**
Related requirements	NFR-1, FR-7
Description	Search result presented within an average of 3000 milliseconds
Requirement type	Non-Functional
Test steps	1. Search for a job
Expected result	The application loads the matching jobs within an average of 3000 milliseconds.

Identifier	**NFR-4**
Related requirements	FR-1, FR-2
Description	Credentials are stored safely
Requirement problems	Dependent on a specific type of implementation.
Requirement type	Non-Functional
Test steps	1. Try to access user credentials.
Expected result	User credentials are not obtainable.

3. System Interfaces

The application should be implemented as a web application utilizing external APIs to fetch data. Thus, the application mainly needs support for common web protocols such as HTTPS over port 443 and HTTP over port 80. The application is intended to be implemented as a single page application which requires the end user to have support for JavaScript in the browser.

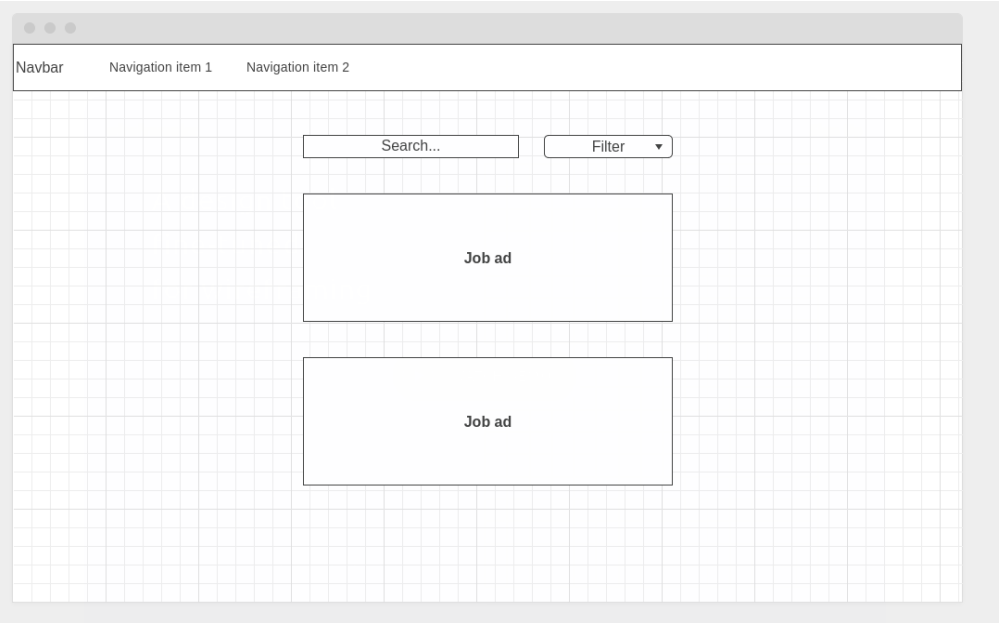
3.1. User Interfaces

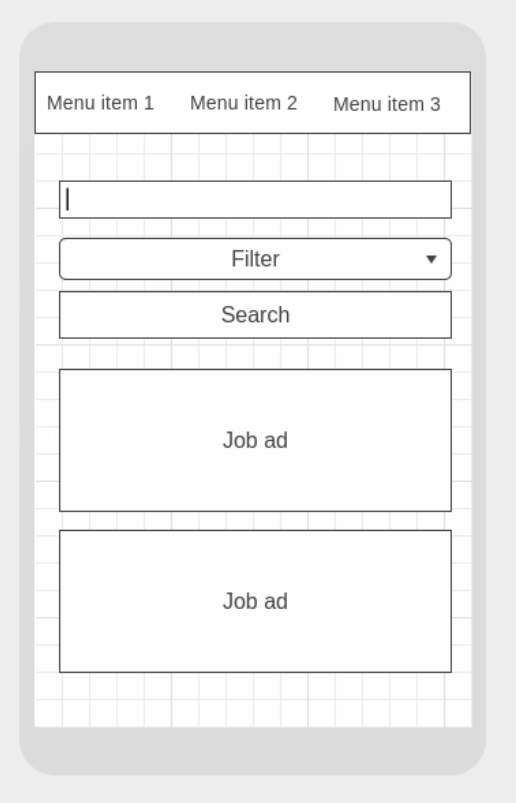
The application should contain a main window and pages for logging in and registering a new user. There must be a view for saved job ads as well.

3.1.1. Look & Feel

The user interface shall adhere to modern conventional design and layout so that the user feels that the interface is intuitive to use. Please refer to sub chapter 3.1.2 Layout and Navigation Requirements for examples.

3.1.2. Layout and Navigation Requirements





The centre of the screen should be occupied by the job ads. The top of the screen should contain the primary navigation together with search and filter components, please refer to section 4.1.2 for more details regarding the filters.

Wireframe prototypes have been presented above, which gives an indication of how the application will look and feel when viewed using a standard desktop browser and mobile device.

3.1.3. Consistency

Naming buttons and other interactable components after their purpose will help enable the users to predict what will happen. By adding visual feedback to the users' interactions, the user gets distinct visual feedback on the actions performed by the system.

A color scheme to provide consistent branding exists, which the web designers have to adhere to. Colors and hex codes:

- #0D3B7A
- #226CCA
- #EDEDED
- #FFFFFF
- #F1843F

3.1.4. User Personalization & Customization Requirements

As the application allows users to search by keywords, the user interface will change depending on what keyword that has been entered in order to view job ads that adhere to the criteria provided by the user.

3.2. Interfaces to External Systems or Devices

3.2.1. Software Interfaces

The application should handle HTTP requests from end-users to the frontend of the application, and REST API calls between the backend and the three primary API endpoints:

- Platsbanken
- Remotive
- RemoteOK

REST API - JSON data transferred internally from backend to frontend.

3.2.2. Hardware Interfaces

This application has no specified hardware requirements.

3.2.3. Communications Interfaces

The application should have support for traffic over port 443 (HTTPS) and port 80 (HTTP). In essence, the application should fully support TCP/IP protocol traffic.

In addition to these, the application will interface with cloud services, in particular, MongoDB Atlas to handle persistent storage. The team will evaluate different cloud service providers to find the most suitable solution for hosting the application's frontend and backend. The decided-upon solution will be presented in further detail during the design phase.

4. Business Rules

4.1. Filtering - FG

4.1.1. Keyword search FG-1

If the user enters a keyword in the search field, then only job ads that match the keyword get presented to the user.

4.1.2. Predefined filters FG-2

If the user selects one or more predefined filters, then only job ads conforming to these filters get presented to the user. At least the following predefined filters should be available:

- Region
- Municipality
- Remote

4.2. User Interface - UI

4.2.1. Restrict the number of ads per request UI-1

The system should implement pagination by the use of “infinite scrolling”, i.e., ads will be loaded when the user scrolls through the search results. By default, a maximum of 20 ads should be loaded per request.

4.2.1. Loading Symbol UI-2

If the user has performed a search and ads have not yet been displayed, then the user shall be presented with a loading symbol indicating that ads are being processed.

4.3. API

4.3.1. API unreachable API-1

If the API for any reason cannot be reached, the user shall be informed about an internal server error.

4.4. Ads AD

4.4.2. Ad redirection AD-1

If a user clicks an ad, then the user shall be redirected to the source provided by the API.

4.5. Authentication/Authorization AA

4.5.1. Anonymous users cannot save ads. AA-1

If a user has not authenticated, then they will not be authorized to save ads to their list of favorites.

4.5.2. Anonymous users cannot log out. AA-2

If a user has not authenticated, then they cannot access the functionality to log out.

4.5.3. Save ads. AA-3

Users can see a list of only their personally saved ads only if they are authorized.

4.5.4. Unique Users. AA-4

If a user tries to sign up using an already existing identifier, then the user must be asked to choose another identifier.

4.5.5. Password Length. AA-5

If a user tries to sign up, they must enter a password that is at least eight characters long.

5. System Constraints

The following development tools have been mandated and will be adhered to:

- Visual Studio Code
- Gitlab

The following implementation tools have been mandated and will be adhered to:

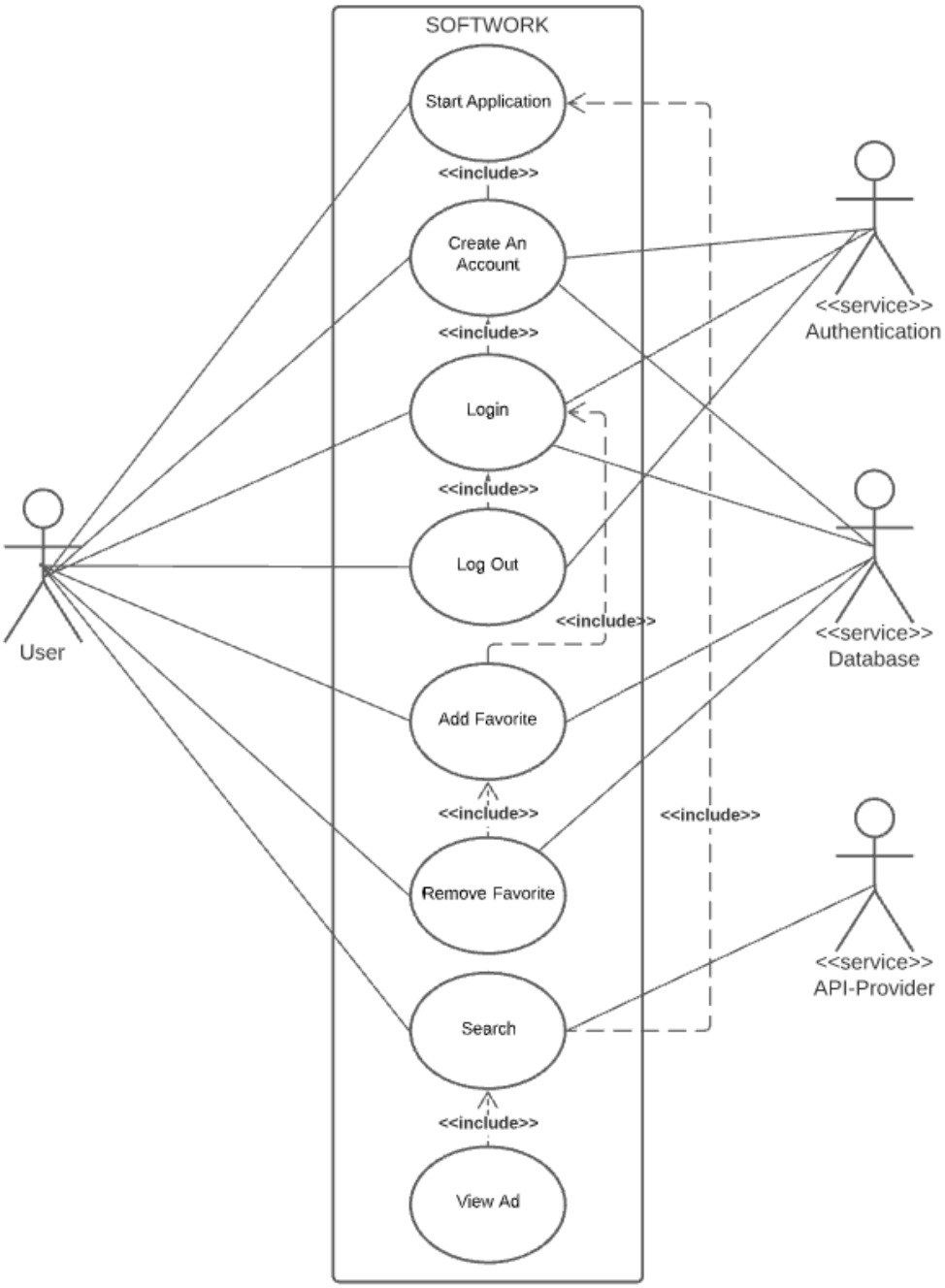
- JavaScript - Will be used as the default implementation language.
- Vue.js - Will be used as a frontend framework.
- Node.js - Will be used as a backend framework.
- Express.js - Will be used as a web server framework.
- MongoDB (NoSQL) - Will be used as the primary database.

The following platform support has been mandated and will be adhered to:

- All modern browsers will be supported on the client-side
- The system will have cross-platform support

There will be no hardware/resource limitations since the software is intended to be deployed using a third-party cloud provider. This means that the application can be scaled to meet the demands of an ever-growing target market.

6. Use-Cases


SOFTWORK
Start Application
<<include>>
Create An Account
<<include>>
Login
<<include>>
Log Out
<<include>>
Add Favorite
<<include>>
Remove Favorite
<<include>>
Search
<<include>>
View Ad
User
<<service>> Authentication
<<service>> Database
<<service>> API-Provider
<<include>>
<<include>>
<<include>>

6.1. Use-Case: Start application

6.1.1. Brief Description

User visits application.

6.1.2. Actor Brief Descriptions

The user that uses the application.

Client application.

6.1.3. Basic Flow of Events

1. The user chooses to start the application.
2. The main view is presented to the user

6.1.4. Alternative Flows

2.1. An error has occurred

1. The system presents an error page

6.1.5. Post-conditions

The main view has been rendered by the system.

6.1.6. Special Requirements

NFR-2 The application should be usable on both desktop and mobile.

6.1.8. Key Scenarios

6.1.5. An error has occurred

6.2. Use-Case: Create an account

6.2.1. Brief Description

The user registers an account of their own in the application.

6.2.2. Actor Brief Descriptions

The user that intends to register an account

Client application

6.2.3. Preconditions

6.1 Start Application

6.2.4. Basic Flow of Events

1. The user selects “Register” in the navigation bar
2. The system presents a registration form
3. The user enters their email address and password
4. The system asks the user to enter the password a second time.
5. The user enters the password a second time.
6. The system checks that email address is valid and passwords match.
7. The user submits the form.

6.2.5. Alternative Flows

6.1 Invalid Email Address.

1. The system displays a message indicating the email address is invalid.

2. The use case resumes at step 3.

6.2 Passwords do not match.

1. The system displays a message indicating the passwords do not match.
2. The use case resumes at step 5.

7.3 User account already exists.

1. The system displays an error message that the account already exists.
2. The use case resumes at step 2.

7.4 No response from the backend.

1. The system displays an error message that the backend cannot be reached.
2. The use case resumes at step 5.

6.2.6. Post-conditions

A new account has been created with the provided credentials.

6.2.7. Special Requirements

NFR-4 User credentials must be stored safely.

6.2.8. Key Scenarios

9.2 No response from the backend.

6.3. Use-Case: Login

6.3.1. Brief Description

User logs in to an existing account.

6.3.2. Actor Brief Descriptions

The user that uses the application.

Client application.

6.3.3. Preconditions

6.2 Create an account

6.3.4. Basic Flow of Events

1. The user selects “Login” in the navigation bar.
2. The system presents the user with a login form.
3. The user submits its credentials.
4. The system validates the user.
5. The user gets logged in.

6.3.5. Alternative Flows

3.1. User enters wrong credentials

1. The system displays an error message indicating that the credentials are incorrect.
2. The use case resumes at step 2.

3.2 No response from the backend.

1. The system displays an error message that the backend cannot be reached.
2. The use case resumes at step 2.

6.3.6. Post-conditions

The user gets logged in.

6.3.7. Key Scenarios

6.3.5 User enters wrong credentials

6.4. Use-Case: Log out

6.4.1. Brief Description

The user logs out from the application.

6.4.2. Actor Brief Descriptions

The user that uses the application

Client application

6.4.3. Preconditions

6.3 Login

6.4.4. Basic Flow of Events

1. The user selects “Log out” in the navigation bar.
2. The system logs the user out.

6.4.5. Post-conditions

The user gets logged out from the application.

6.5. Use-Case: Add Favorite

6.5.1. Brief Description

The user wants to save a job ad.

6.5.2. Actor Brief Descriptions

The user that uses the application.

Client application.

6.5.3. Preconditions

6.3 Login

6.5.4. Basic Flow of Events

1. The user observes a job ad they like and selects the “Add favorite” choice.
2. The system adds the ad to the user’s personal list in the database.

6.5.5. Post-conditions

The selected job ad has been added to the user’s personal favorite list.

6.5.6. Alternative Flows

2.1 No response from the backend or database.

1. The system displays an error message that the backend cannot be reached.
2. The use case exits.

6.6. Use-Case: Remove Favorite

6.6.1. Brief Description

The user wants to remove a job ad from his/her list of favorites.

6.6.2. Actor Brief Descriptions

The user that uses the application.

Client application.

6.6.3. Preconditions

6.5 Add Favorite

6.6.4. Basic Flow of Events

1. The user observes an ad in their personal favorites list. They select the "Remove favorite" option.
2. The application removes the job ad from their list.

6.6.5. Post-conditions

The selected job ad has been removed from the user's personal favorite list.

6.6.6. Key Scenarios

6.6.5. The actor refuses to confirm

6.7. Use-Case: Search

6.7.1. Brief Description

The user searches for job ads using keywords and predefined values.

6.7.2. Actor Brief Descriptions

The user that uses the application.

Client application.

6.7.3. Preconditions

6.1 Start application

6.7.4. Basic Flow of Events

1. The user searches
2. The application displays the search results

6.7.5. Alternative Flows

1.1 User searches without choosing a predefined value or any keywords

1. The system displays a message indicating that an empty search cannot be performed
2. The use case resumes at step 1

2.1 No job ads exist for the search parameters

1. The system displays a message indicating that no results were found
2. The use case resumes at step 1

6.7.6. Sub flows

User searches using user-defined keywords

1. The user searches for a specific keyword
2. The system displays the search results (step 2)

User searches using predefined values

1. The user selects predefined criteria and submits a search

2. The system displays the search results (step 2)

User searches using both predefined values and user-defined keywords

1. The user selects a predefined value and enters keywords
2. The user performs a search

6.7.7. Post-conditions

The system presents the user with relevant job ads

6.7.8. Special Requirements

NFR-3 Search results should be presented within an average of 3000 milliseconds.

6.7.9. Key Scenarios

6.7.4. Main Scenario

6.8. Use-Case: View Ad

6.8.1. Brief Description

The user wants to view an ad presented by the search result.

6.8.2. Actor Brief Descriptions

The user that uses the application.

Client application.

6.8.3. Preconditions

6.7 Search

6.8.4. Basic Flow of Events

1. The user clicks on the job ad he/she wishes to view
2. The application redirects the user to the job ad

6.8.5. Post-conditions

The system redirects the user to the job ad.

7. Performance modelling

As we do not have any real data to model yet, a lot of assumptions have been made. In order to get a scenario that can be somewhat comparable to a real-life case, we ran a simulation based on the assumption that the system will serve at least 25 000 users in a single day. Hence, we can derive the following maximum values for our queueing network model:

T = 86 400
C = 25 000

With these assumed attributes, we can calculate the minimum required throughput of the system (X = C / T), which in this case is equal to 0.29.

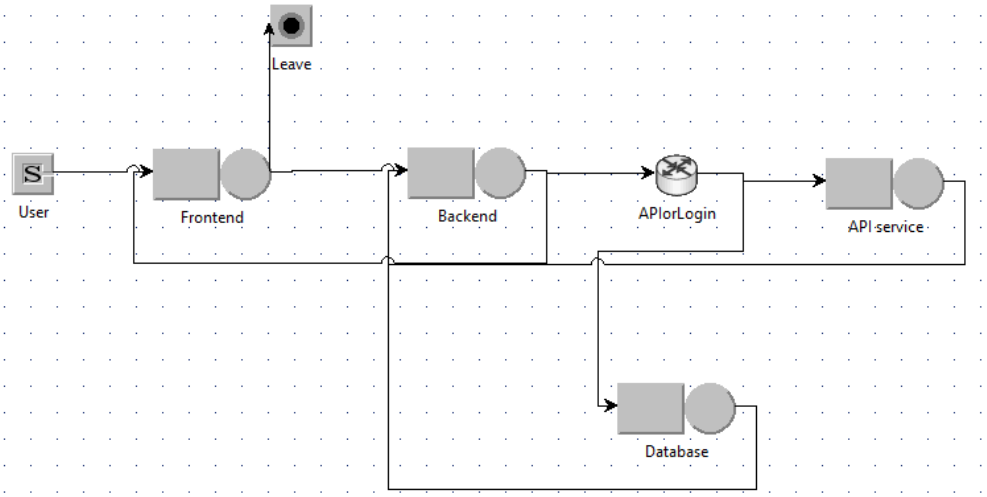
Each user session started executing in the Frontend whose service time is assumed to be 300ms on average. After execution, the flow continued to two possible paths: leave or proceed to the backend. We assume that 30% choose to leave.

From the Frontend, the user may search for a keyword or choose to log in; both options will be handled by the backend and we have assumed that 25% chose to login and 75% chose to search. We have assumed that the backends' service time is 400ms on average as well.

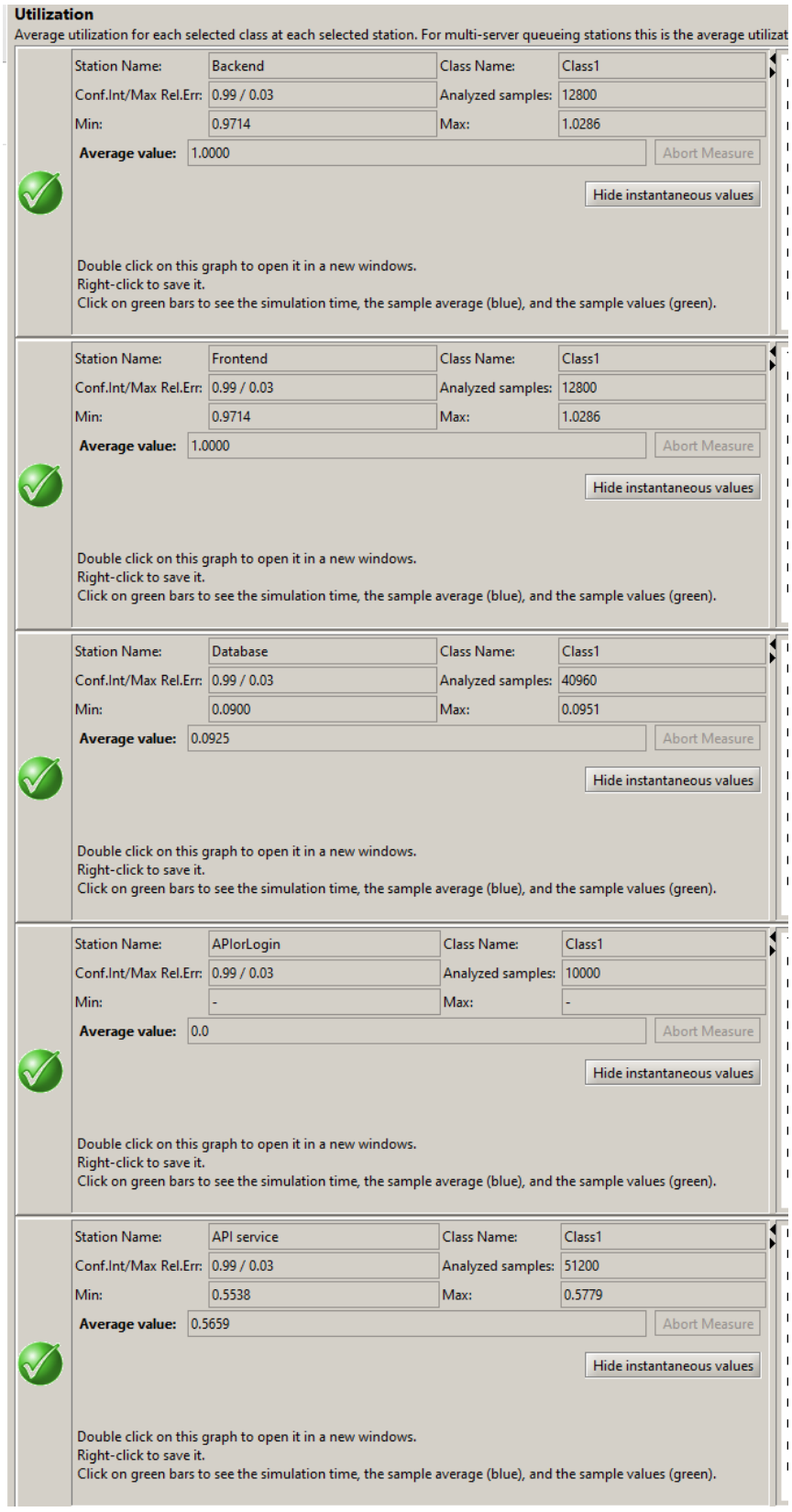
If the user proceeds to log in, the application will interact with the database to fetch the user information. The database is assumed to have a service time of 300ms.

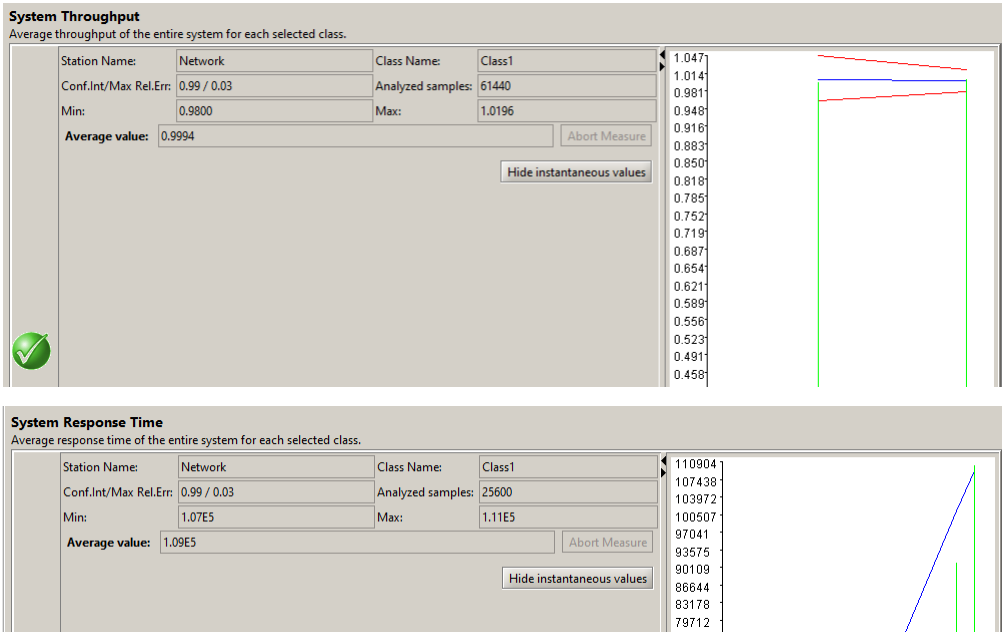
If the user chooses to make a search, the backend will make an API call to an API service. We estimate the response time to be 600ms.

The model of the application.

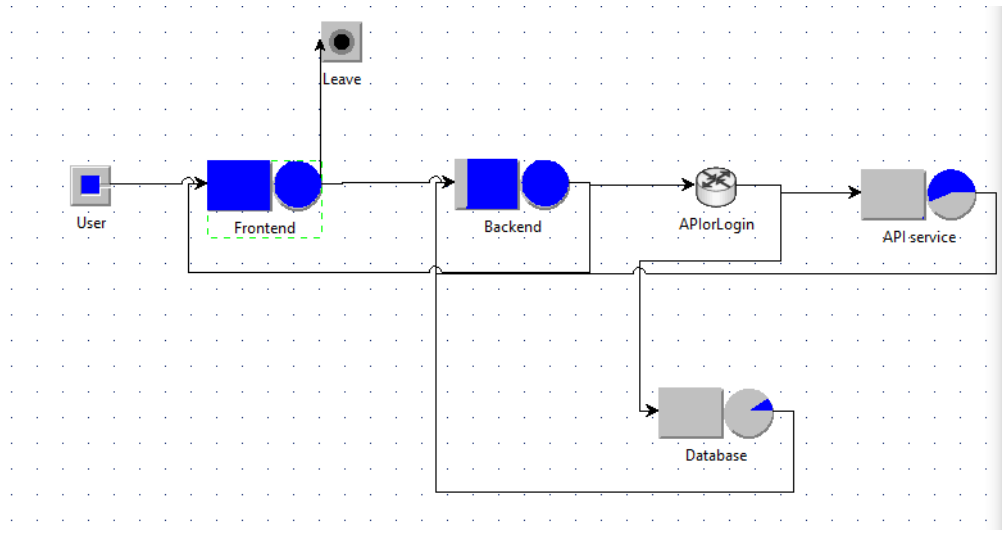


Upon running the simulation, these are the results. The utilization of the services:


Utilization
Average utilization for each selected class at each selected station. For multi-server queueing stations this is the average utilizat
Station Name: Backend
Class Name: Class1
Conf.Int/Max Rel.Err: 0.99 / 0.03
Analyzed samples: 12800
Min: 0.9714
Max: 1.0286
Average value: 1.0000
Abort Measure
Hide instantaneous values
Double click on this graph to open it in a new windows.
Right-click to save it.
Click on green bars to see the simulation time, the sample average (blue), and the sample values (green).
Station Name: Frontend
Class Name: Class1
Conf.Int/Max Rel.Err: 0.99 / 0.03
Analyzed samples: 12800
Min: 0.9714
Max: 1.0286
Average value: 1.0000
Abort Measure
Hide instantaneous values
Double click on this graph to open it in a new windows.
Right-click to save it.
Click on green bars to see the simulation time, the sample average (blue), and the sample values (green).
Station Name: Database
Class Name: Class1
Conf.Int/Max Rel.Err: 0.99 / 0.03
Analyzed samples: 40960
Min: 0.0900
Max: 0.0951
Average value: 0.0925
Abort Measure
Hide instantaneous values
Double click on this graph to open it in a new windows.
Right-click to save it.
Click on green bars to see the simulation time, the sample average (blue), and the sample values (green).
Station Name: APIorLogin
Class Name: Class1
Conf.Int/Max Rel.Err: 0.99 / 0.03
Analyzed samples: 10000
Min: -
Max: -
Average value: 0.0
Abort Measure
Hide instantaneous values
Double click on this graph to open it in a new windows.
Right-click to save it.
Click on green bars to see the simulation time, the sample average (blue), and the sample values (green).
Station Name: API service
Class Name: Class1
Conf.Int/Max Rel.Err: 0.99 / 0.03
Analyzed samples: 51200
Min: 0.5538
Max: 0.5779
Average value: 0.5659
Abort Measure
Hide instantaneous values
Double click on this graph to open it in a new windows.
Right-click to save it.
Click on green bars to see the simulation time, the sample average (blue), and the sample values (green).



As we can see above, the simulated response time of the system is below the time required by NFR-3 (less than 3000ms).



As can be seen in the picture, the frontend and the backend are the most likely candidates to get over saturated. With these metrics, they are both on their limits. If the system is to be scaled up, these two services are the first two services to be scaled up.