Linköping University | Department of Computer and Information Science

Master's thesis, 30 ECTS | Computer Engineering

22 | LIU-IDA/LITH-EX-A--22/043--SE

Development of an evaluation model for client-side JavaScript Frameworks

Utveckling av en utvärderingsmodell för klientbaserade JavaScript-ramverk

Ellen Sundholm
Sebastian Retzius

Supervisor : Jonas Wallgren
Examiner : Kristian Sandahl

External supervisor : Erik Wiström

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Upphovsrätt

Copyright

Abstract

There are many alternatives to choose from when deciding what client-side JavaScript framework to use, and these are growing by the day, which leaves developers with the difficult task of deciding the most suitable tool. Little research has been done on what drives the selection of a client-side JavaScript framework. Web developers at Exsitec AB are at times put in the position to choose which framework to use in a project, and as of today no specific basis for this choice exists. The purpose of this master's thesis was to facilitate the process of choosing a client-side JavaScript framework for web developers, with the target group being web developers at Exsitec. The purpose was divided into two research questions. The first question was intended to find out important evaluation criteria in the choice of a client-side JavaScript framework and the second question was about developing a model for evaluating frameworks based on these important evaluation criteria. A pre-study was conducted by first researching literature about important evaluation criteria in the choice of a client-side JavaScript framework and then conducting interviews with and sending out a survey to web developers at Exsitec. Results from this was then used as a basis for developing an evaluation model that web developers can use in the choice of a client-side JavaScript framework. Firstly, a first draft of the evaluation model was made. Secondly, this model was tested by evaluating three JavaScript front-end frameworks React, Angular and Vue. Thirdly, the evaluation model was refined based on insights during testing. The conclusions drawn in this master's thesis are a list of important evaluation criteria and an evaluation model based on these criteria that aid in understanding whether a client-side JavaScript framework is a good fit in a specific situation.

Acknowledgments

The authors of this master's thesis would like to thank all employees at Exsitec that have helped during the creation of this thesis. Especially thanks to supervisor Erik Wiström. We would also like to thank Kristian Sandahl and Jonas Wallgren at LiU for helping and supporting us during our work. Lastly, we would like to thank Mathilda Moström and Sophie Ryrberg for giving valuable feedback during our work with this thesis.

Contents

List of Figures

List of Tables

1 Introduction

This chapter gives the reader an introduction to this master's thesis. First a motivation is presented in section 1.1, followed by a presentation of the aim of this master's thesis in section 1.2. Research questions are presented in section 1.3. Lastly, the delimitations of this master's thesis are considered in section 1.4.

1.1 Motivation

A common way to build web applications is through the client-server architecture in which the web application consists of a centralized server that many different clients can connect to [7]. A client is what the user interacts with and the server does the logical handling of the data that is sent from different clients when users interact with them [7].

JavaScript is the most used client-side programming language for building web applications. 97.6 % of websites use some form of JavaScript [35]. There are many different ways to build a client with JavaScript: a plain HTML/CSS/JS website, using JQuery and more modern client-side JavaScript frameworks (JSFs) are some of them. Statistics shows that the most used client-side JSFs are React, Angular and Vue [1].

JavaScript is a lightweight programming language that is run on the client-side of websites [2]. JavaScript is used to write scripts that control the behaviour of the website when different events that manipulate a page occur [2], for example when a user clicks a button or when the browser reloads [15]. When JavaScript was introduced it allowed websites, that previously only were static documents, to be interactive with the user [14]. After JavaScript was released developers who were working with JavaScript started sharing solutions to typical problems in so-called shared libraries, which greatly shaped how development for the web evolved [14]. Modern client-side JSFs are a result of this sharing of solutions [14]. They offer structured ways to easily build scalable, maintainable, and highly interactive dynamic web applications [14].

The evolution of client-side JSFs has moved quickly due to the increasing refinement of browser-based applications and the increasing popularity of JavaScript used on servers [36].

There are many alternatives to choose from when deciding what client-side JSF to use, and these are growing by the day, which leaves developers with the difficult task of deciding the most suitable tool [36]. In contrast, the scientific consensus is that too little research has been done on what drives the selection of a JSF and how the selection of a client-side JSF can be done [13, 24, 8].

This master's thesis is done in collaboration with Exsitec which is an IT-consultancy firm. Exsitec sometimes implements custom made projects and in those they are put in the position to choose what front-end framework to use. As of today, this choice is often made based on earlier decisions, and no thorough investigation is conducted before choosing a framework for a project. The difficult decision of selecting a modern client-side JSF presents a need for a better understanding of how to evaluate JSFs and what criteria are important to consider when evaluating them.

1.2 Aim

The aim of this master's thesis is to acquire an understanding of what criteria are important in the evaluation of a JSF and based on this design a model that can be followed by developers when faced with the decision of what JSF to use.

1.3 Research questions

The aim of this master's thesis is divided into two research questions presented below. The first question will give an understanding of what criteria are important when evaluating client-side JSFs. The results from question one will be used as a basis to develop a model for evaluating client-side JSFs, answering research question two.

1. RQ1: What evaluation criteria are important to consider when evaluating whether to adopt a client-side JSF?
2. RQ2: How can a model for evaluating a client-side JSF look like based on the found important evaluation criteria?

1.4 Delimitations

As mentioned earlier, there are many client-side JSFs to choose from and this master's thesis touch on some of these frameworks. However, due to limited time, only a small number of frameworks was examined and experimented on. The frameworks evaluated in this thesis were React, Angular and Vue. The decision of what frameworks to use was taken in collaboration by the authors of this thesis and the supervisor at Exsitec.

Another delimitation in this master's thesis is that the interviewees and survey respondents are all employed at the same company. This due to that this master's thesis was conducted in collaboration with the company Exsitec AB.

The choice of focusing on client-side JSFs instead of other methods for building web clients was partly made in collaboration with Exsitec and partly due to using JSFs being the most popular way of building the front-end of modern web applications.

2 Background

This chapter serves to present the reader with relevant background information. Firstly, a background on how web applications are structured is presented in section 2.1. Secondly, frameworks for building JavaScript clients are presented in section 2.2. Lastly, the company Exsitec who this thesis is done in collaboration with is presented in section 2.3.

2.1 Web application architecture

As stated in the section 1.1, a popular architecture for web applications is the client-server architecture [7]. Reese [28] describes the client-server architecture as the server being the provider of a service and the client making requests to the server and presenting the results returned from the server. The author presents two different versions of this architecture, 2-tiered and 3-tiered. Reese describes the 2-tier architecture as consisting of a database and a client which is directly connected to the database. In this type, the client presents the data and handles the logic of changing data while the database stores the data. He describes 3-tier as consisting of an extra application server which handles the logic that is handled by the client in the 2-tiered architecture. The server acts as a middleware between the client, which now only presents information, and the database. Since the logic is handled by the server, this architecture allows for clients that need less resources to run, also known as thin clients. To communicate between the client and the server, Hypertext Transfer Protocol is used. Nielsen et al. [22] describe Hypertext Transfer Protocol, or HTTP, as a protocol used for transmitting documents between a client and a server on the web. Nielsen et al. state that, in HTTP, the client initiates the transfer by sending a request to the server containing information about what the client is requesting. The server then responds with the requested information. In this master's thesis, a 3-tiered architecture where the client communicates with the server using HTTP is implemented.

2.1.1 Single Page Applications

The clients that are implemented in this master's thesis are single page applications, or SPAs. An SPA is a web application which does not always have to reload the page from the server when the web application changes [17]. Instead, this is done by dynamically updating the

current page and requests to the server are only sent if additional resources are needed [16]. A full HTML-page is created and this page is then dynamically rewritten, instead of how it is done with traditional multi page applications, MPAs, where the web application has to load entire new HTML-pages from the server. SPAs are more responsive and are perceived as faster than traditional MPAs as well as more user-friendly [16]. An example of an SPA architecture is shown in figure 2.1.

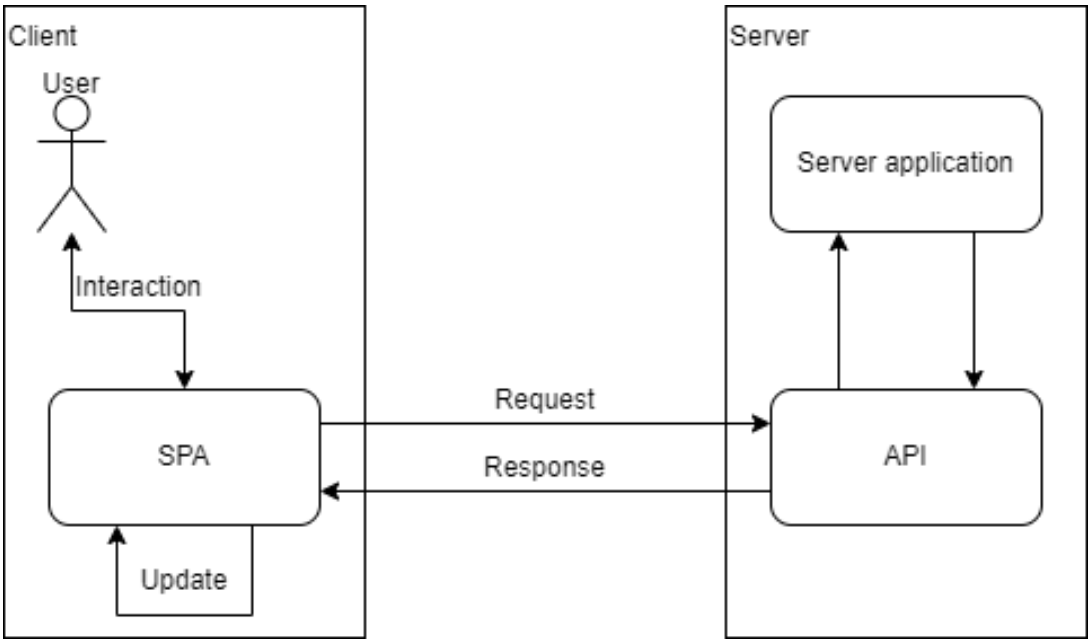

Figure 2.1: Example of SPA architecture where user interaction can trigger an update of the SPA and/or a request to the server

2.1.1.1 Routing

Routing handles what is shown by the SPA when a user visits a certain URL route. *https://my-app/home* and *https://my-app/users/2* are examples of URL routes that would be handled differently by the routing tool of an SPA. The app senses what URL route is given and renders the page according to which view corresponds to the current route. For example, *https://my-app/home* will render the home view and *https://my-app/users/2* will render a view that shows the user with id 2. Often certain routes are not supposed to be shown for everyone. For example, some routes might only be available for users logged in to the application. This protection of routes can often be implemented with SPA routing tools. [21]

2.1.1.2 Data-binding

Data-binding is a very important part of SPAs. It can be described as a link between the logic and the visual parts of the application. When it comes to SPAs there are two types of data-binding, one-way and two-way data-binding. [38]

One-way data-binding is when data is sent in one direction. It means that when the data is updated, the view is also updated, but not the other way around. On the other hand, two-way data binding is when data can be sent in both directions. Changes in the view update the data and changes in the data update the view. [12]

2.1.1.3 State

Managing states is an important part of a single page application. When it comes to SPAs, a state could be described as a data structure which contains information about the application. States can be divided into UI states and global states. UI states describe the visual aspects of the application whereas global states are data that needs to be stored and exsist between different parts of the application and that can be restored when the application restarts. [32]

2.2 Client-side JavaScript frameworks

As mentioned in the motivation section 1.1, JSFs are shared libraries that provide a method to build web clients in a way which make them highly interactive and easy to scale up and maintain [14]. In this section, three of the most popular frameworks [1] for building SPAs are presented. A short description and background of each framework is presented as well as a description of how the frameworks handle HTTP-requests, states, routing and data-binding.

2.2.1 React

Uzayr et al. [36] describe React[1] as a JavaScript library that is used to build user interfaces and that it is maintained by Facebook and a community of companies and developers. They state that React technically is not a framework. However, due to its popularity and that it is component based, the library is often compared and mentioned in connection with other JSFs. React only supports one-way data binding [27]. React does not have any built in tools for HTTP-requests, routing and state-handling. However, there are third-party libraries for handling these tasks. React Router[2] is the most popular library for handling routing and Redux[3] is the most popular library for state-handling. Redux also works for any JavaScript app. There are multiple libraries for handling HTTP-requests in JavaScript which can be used in React, some examples being JavaScript's built in Fetch API[4], AJAX[5], and Axios[6]. A more complex React specific tool is React Query[7].

2.2.2 Angular

Angular[8] is a framework for building front-end web applications that is developed and maintained by Google and a community of volunteers according to Uzayr et al. [36]. They describe Angular as a module-based approach for building apps with components that is based on Typescript, which is a strongly typed extension to JavaScript. Angular supports both one-way and two-way data-binding [34]. HTTP-requests are handled by the HTTP client built into the Angular framework and Angular Routing is a tool built into the Angular framework that handles routing. NgRX[9] is a framework for building reactive Angular applications which offer solutions for state-handling with global and local states.

2.2.3 Vue

Uzayr et al. [36] describe Vue[10] as a progressive JavaScript client-side framework that is implemented as an extra markup to HTML. Uzayr et al. say that the central library consists of

[1] https://reactjs.org/
[2] https://reactrouter.com/
[3] https://redux.js.org/
[4] https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
[5] https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started
[6] https://www.npmjs.com/package/axios
[7] https://react-query.tanstack.com/
[8] https://angular.io/
[9] https://ngrx.io/docs
[10] https://v3.vuejs.org/

the "view" layer which can be integrated with other libraries or projects. Vue can also be used to create full single page applications. Uzayr et al. state that Vue is easy to use and is seen to have an easier learning curve than many other similar frameworks, something that has made the framework very popular in recent years. Vue offers solutions for both one-way and two-way data-binding [9]. Vue Router[11], is the official router for Vue applications. Vue offers state-handling in the form of Vuex[12] or Pinia[13], both of them being state management libraries who handle the states throughout the Vue application [31]. Just as with React there exists many options to choose from for making HTTP-requests in Vue and the Vue documentation recommends using Axios[14] or Fetch API[15].

2.3 Exsitec

This master's thesis is conducted in collaboration with Exsitec[16], an IT-consultancy firm that offers solutions for integration, digitization and system support. Developers at Exsitec are sometimes put in the position of choosing an appropriate front-end framework for a project. Today, this decision is not based on any evaluation criteria. The framework that most employees are comfortable with, which is Angular, is often chosen without a proper evaluation of alternatives. Also, a starter application which can be used as a basis to continue to develop on, though being old, exist in the Angular framework. This further influences the decision towards choosing Angular. The fact that Angular often is chosen without a thorough investigation opens up for a need for more routine when it comes to the choice of a front-end framework, and especially a JSF, for a project. As well as a better understanding of what criteria are important in the evaluation of a JSF.

[11] https://next.router.vuejs.org/
[12] https://vuex.vuejs.org/
[13] https://pinia.vuejs.org/
[14] https://www.npmjs.com/package/axios
[15] https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
[16] https://www.exsitec.se/

3 Related Work

This chapter serves to present the reader with related work by other authors. Related work regarding what criteria other authors have used to perform evaluations of JSFs is presented in section 3.1. Section 3.2 presents related work concerned with the evaluation of JSFs. Section 3.3 presents evaluation models developed by other authors.

3.1 Evaluation of adoption criteria

Graziotin & Abrahamsson [13] criticised the, at the time (2018), existing literature regarding comparison of JSFs. They argued that the focal point of previous research being on software metrics and benchmarks was diverging from the interest of practitioners. The authors highlighted the value of not only regarding academic research when comparing JSFs, but to combine it with insights about why JSFs are chosen from the practitioner's view. Therefore, the authors developed a comparison model for helping in the selection of a JSF. The model they developed builds on the comparison by Gizas et al. [10], based on quality, validation and performance. Graziotin and Abrahamsson conducted interviews with developers to find out what practitioners find important. Based on the results from the interviews, Graziotin and Abrahamsson added three new factors to the comparison model: adequacy of documentation, community participation and pragmatics. Pragmatics is described as code less, do more factors.

Pano et al. [24] continued to build on the work of Graziotin and Abrahamsson [13] by examining factors and actors that affect the choice of a JSF. In order to do so, Pano et al. conducted a study based on interviews with 18 people who were involved in choosing JSFs and could motivate why they chose a specific one. The authors developed a model which categorizes desirable features under the five categories: performance expectancy, effort expectancy, social influence, facilitating conditions, and price value. The features themselves and their categories can be seen in table 3.1. They also presented the customer, developer, team and team leader as the actors affecting the adoption of a specific framework.

Performance expectancy was described by Pano et al. [24] as the combination of performance of the final application in terms of memory and bandwidth consumption as well as the size

Table 3.1: Categorized features presented by Pano et al. [24]

Category	**Features**
Performance expectancy	Performance Size
Effort expectancy	Automatization Learnability Complexity Understandability
Social influence	Competitor analysis Collegial advice Community size Community responsiveness
Facilitating conditions	Suitability Updates Modularity Isolation Extensibility
Price value	Cost

of the code needed to perform basic functionality. The authors described effort expectancy as the ease of use associated with the framework as well as aspects that simplify programming tasks. They put the features automatization, learnability, complexity, and understandability under this category. Social influence was described by the authors as consisting of competitor analysis, collegial advice (recommendations from peers), community size (in terms of the number of people contributing framework), and community responsiveness. Facilitating conditions were, according to the authors, suitability for the task at hand, updates to the framework, modularity, isolation, and extensibility. Lastly, the authors stated that the price of the framework is a part of the choice, where free frameworks are often favored.

Ferreira et al. [8] conducted a study to understand reasons for both adoption and un-adoption of JSFs, focusing on React, Angular, Vue, Backbone, and Ember. The authors did this by surveying prominent front-end developers for the most popular projects on GitHub. In their study they identified nine factors that are used to motivate adoption of a certain framework and ranked them by occurrences in the survey responses, see table 3.2.

Table 3.2: Ranked factors by Ferreira et al. [8]

Rank	**Factor**	**Description**
1	Popularity	This framework is widely known and used
2	Learnability	This framework is easy to learn and use
3	Architecture	This framework forces client to follow a solid architecture
4	Expertise	I have previous expertise in this framework
5	Community	This framework is maintained by a large community
6	Performance	This framework has excellent performance
7	To gain experience	I wanted to gain experience in this framework
8	Documentation	This framework's documentation is very good
9	Sponsorship	This framework is supported by well-known companies

Ferreira et al. [8] also recommended that when adopting a framework one should evaluate the available documentation for the framework, analyze whether the intended application architecture fits with the, sometimes, enforced architecture of a framework and to analyze the expertise regarding the framework within the team.

Molin [20] conducted a master's thesis where he aimed to formulate a way to scientifically compare JSFs. To find relevant criteria to base the comparison on he reviewed literature on the subject as well as conducted interviews with developers. After cross-analyzing the criteria found in literature and the criteria mentioned in interviews Molin settled for nine criteria which can be found in table 3.3.

Table 3.3: Criteria found by Molin [20]

Criteria	**Description**
Security	Security against attacks resulting in user data being stolen
Modularity	The structure of the app supports low coupling of components
Popularity	Many people are using the framework
Maturity and Stability	The framework is used by large corporations
Simplicity and Usability	It's simple to extend the framework and there is good support (for example good documentation)
Portability and Compatibility	The framework shall work for many different browsers and devices
Cache Performance and Persistence	Apps built with the framework executes quickly and is fast to update
Testability	The framework allows for proper testing according to standards
Maintainability	It's easy for a company to maintain an app developed in this framework

The studies by Graziotin and Abrahamsson [13], Pano et al. [24] and Ferreira et al. [8] indicate that when it comes to the adoption of a front-end framework, it is important to evaluate more than just software metrics. It is important to analyze the practitioners view and examine what are important criteria for them as well. This is relevant to this master's thesis since this shows the importance of examining what are important criteria for the people that are decision-makers in the adoption of a JSF. These articles also give inspiration regarding what evaluation criteria can be used in this thesis.

3.2 Evaluation of JavaScript frameworks

One of the first studies that evaluates client-side JSFs was done by Gizas et al. [10]. They conducted an evaluation of client-side JSFs and compared these in terms of performance and software quality. The evaluation was done on six of the, at that time, most popular client-side JSFs: ExtJS, Dojo, jQuery, MooTools, Prototype, and YUI. The JSFs differed in what they offered resulting in that Gizas et al. defined a core version with a set of functions and created this for each of the JSFs, to be able to evaluate and compare these properly. Quality tests were conducted by evaluating size, complexity, and maintainability of the code using software metrics. To be able to evaluate performance, a test framework called SlickSpeed selectors[1] was used, where the JSFs were tested in four different operating systems and seven browsers. The frameworks were evaluated based on execution time on the different operating systems and browsers. On top of evaluating performance and software quality, the authors also conducted validation testing on the six frameworks. This was done by combining the use of Yasca[2] software utility and JavaScript Lint[3] and evaluating what kind of errors the JSFs had, divided into critical, high severity, and overall errors. Gizas et al. did not conduct this evaluation with the intended result to find the best JSF, but focused more on highlighting drawbacks and advantages between the different frameworks.

[1] `https://github.com/kamicane/slickspeed/tree/master`
[2] `https://sourceforge.net/projects/yasca/`
[3] `https://www.javascriptlint.com/`

Delčev and Drašković [6] conducted a survey study on, at the time of their study (2018), modern client-side JSFs, where they compared the frameworks AngularJS, Knockout, Backbone and Ember. This comparison highlighted drawbacks and benefits with the different frameworks, as well as analyzed and compared the frameworks regarding performance and security. Performance was evaluated based on the frameworks' sizes and operations per second and Delčev and Drašković results showed that AngularJS had the best performance of the compared frameworks. Retire.JS was used to discover vulnerabilities in the frameworks, showing that the most vulnerabilities were connected to the templates. On top of evaluating performance and security, the frameworks were also compared regarding documentation, architecture, web browser compatibility, technical possibilities, learning time, and complexity. Delčev and Drašković conclude that Backbone differs a lot from the other frameworks since more decisions can be taken by the developer, whereas the other comes with much more finished functionalities. The study also showed that all frameworks were similar in the sense that they follow a clear structure leading to more readable code. The survey study showed that all frameworks had negative sides, these being associated with security, testing, technical documentation, or execution speed.

Mariano [19] conducted a master's thesis project where React, Angular and BackboneJS were evaluated and compared to each other. Mariano's evaluation was based on benchmarks on performance of the applications built in each framework and software metrics of the code for these applications. The author developed two different applications in each framework and then conducted benchmarks for the frameworks on lines of code, cyclomatic complexity, halstead complexity, maintainability index, database metrics, page-load/rendering time, speed index as well as render and frame measurements.

Ivanova and Georgiev [16] conducted a study on modern client-side frameworks, using a practical approach, to find the most suitable client-side framework for developing an education platform. The aim of the study was to give the reader a better understanding of the popular concepts and techniques that exist regarding web development, as well as to compare and analyze modern client-side frameworks. The frameworks/libraries that were compared by Ivanova and Georgiev are React.js, Angular and Vue.js. The frameworks were compared and evaluated regarding directives and components, programming language, templates, size and performance, testing, and learning curve. Based on results from the comparison of the frameworks, the authors decided to develop the education platform with Angular since this framework was seen as most suitable.

Molin [20] evaluated AngularJS, Angular and React based on the criteria he found important that was presented in section 3.1. Molin stated that AngularJS was the best framework at the time of his study (2016) due to the three frameworks performing similar in most criteria, but AngularJS being the more mature and popular framework.

Gizas et al. [10] and Marino [19] conducted evaluations of JSFs based on different performance metrics and classical software quality metrics. Delčev and Drašković [6], Ivanova and Georgiev [16] and Molin [20] conducted studies where other criteria, which are not focused on the output (the code and the product), are used in combination with performance and software metrics. The added criteria are connected to the development process, such as the ones brought up in section 3.1. These articles all give inspiration on how JSFs can be evaluated. Especially interesting are the article by Ivanova and Georgiev [16] and the master's theses by Mariano [19] and Molin [20] since they all use a practical approach in their study, using similar or the same client-side frameworks as this thesis intend to evaluate. This is relevant to this master's thesis since web applications in different frameworks will be developed and inspiration regarding how this study is conducted can be taken from the studies.

3.3 Evaluation models

In this section different approaches for the construction of an evaluation model are presented. Firstly, models where the user evaluates the evaluation object based on predetermined questions are described. Secondly, an analytical hierarchy process, which compares different objects based on evaluation criteria to give the answer of which is best suitable for the current situation, is presented. In this section the focus is on presenting different evaluation models and therefore some of the presented related work does not touch on JSFs. However, they present evaluation methods that can act as inspiration to the JSF evaluation model in this master's thesis.

3.3.1 Question Based Models

Kaluža et al. [17] analyzed different client-side JSFs to be able to compare these when it comes to which are better suited for development of SPAs and MPAs. The evaluated frameworks in the study are Angular, React and Vue. These were chosen by the authors since their own research showed that these were the most popular JSFs at the time (2018). The evaluation was conducted by analyzing frameworks based on questions, derived by the authors, which were related to development of SPAs and MPAs. The three frameworks were evaluated based on these questions, both with results as text (yes/no) and given points based on how suitable they were for development of SPAs or MPAs (0, 1, 1.5 or 2). The results were then summarized and analyzed by the authors, leading to an indication to which framework is best suited for SPAs and which framework that is best suited for MPAs.

Thesing et al. [33] conducted a study on how a decision model for deciding the appropriate project management approach for a specific project can be developed. The authors focused on a waterfall approach vs an agile one and tried to map advantages, disadvantages and differences with these two approaches. Thesing et al. researched the subject by an analysis of relevant literature to get insights, which then were compared with insights from 15 expert interviews. This research was used to develop a decision model based on 15 criteria that were found important when evaluating which project management approach to choose. These 15 criteria are sub-critera of five categories: scope, time, costs, organisation context, and project-team characteristics. The presented model consists of two steps, the first being examination of exclusion criteria and the second being scoring of the 15 characteristics. In the first step of the model, exclusion criteria are evaluated regarding the project. If any of the exclusion criteria apply, the conclusion is that the Waterfall approach is best suited for the project. If no exclusion criteria apply, the model continues, and the 15 criteria are scored based on a grading scale. Thesing et al. used the following scoring in their study:

- (4) characteristic is fully applicable
- (3) characteristic applies to a large extent
- (2) characteristic is partially applicable
- (1) characteristic applies only to a very limited extent
- (0) characteristic does not apply.

The criteria are formulated so that they represent if the project is suited for a waterfall approach or not. A high score means that the waterfall approach is a good approach for the concrete project whereas a low score indicates that an agile approach is better suited. Thesing et al. included the possibility to add weight to the different criteria or categories in their model. These weights can be adjusted based on the concrete project characteristics. The

authors gave recommended weights based on their research and conducted interviews on which criteria that were perceived to be most important. The proposed weights had a grade from 0 to 3, with 0 being 'Low importance' and 3 being 'High importance'.

Molin [20] used a similar model to Thesing et al. and Kaluža et al. that is based on answering questions to compare the frameworks examined in his master's thesis. Molin's questions were mostly answered with yes/no answers or simple facts, for example the answer to who develops each framework was simply the corresponding company (Facebook Inc. now Meta Platforms, Inc. for React and Google Inc. for AngularJS and Angular). These answers did not allow for further analysis between the frameworks, rather the answers only answers if a framework fulfills the question or not.

Inspiration can be taken from all these articles, [17, 33, 20], and their presented methodologies when designing the JSF evaluation model. Kaluža et al., Thesing et al. and Molin all base their evaluation models on answering predetermined questions with numerical scoring and/or yes/no questions and Thesing et al. introduce a weighting system for placing emphasis on specific criteria.

3.3.2 Analytical Hierarchy Process

Another possible approach to evaluate the JSFs is Analytical Hierarchy Process, or AHP. AHP was described by Brunelli [4] as a theory and mehod for evaluating alternatives relative to each other. The author described that this is suitable when the proportions between the measures are interesting but not the actual measures. AHP is based on comparing different options pairwise for a decision, and that the decision itself is divided into smaller sub-problems which can be considered easier to handle. When using AHP the different criteria (sub-problems) can have different effects on the overall goal, and therefore these can be weighted in relation to each other [4, 30]. Brunelli [4] described an example that explains how AHP can be used when deciding a destination for a holiday, where three cities are presented as alternatives. The overarching goal is a satisfactory holiday for the whole family, and criteria considered to have an impact on the overall satisfaction is the destination's climate, environment, and cost. The environment is considered to be the most important of these and therefore is given the most weight. The destinations are then compared pair-wise within each criterion where they are judged proportionate to how much better one alternative is compared to the other. The results are then standardized and compiled mathematically so that the alternatives are ranked within each criterion. Based on the weights of the different criteria the rankings within each criterion can be compiled into a final overall ranking, where the highest ranked alternative is considered to be the best.

Two examples of how AHP is used is described by Wei et al. [37] as well as Pandey & Litoriya [23] who evaluated enterprise resource planning systems and agile processes in web application development respectively. Wei et al. described how AHP works as well as a real-world case of deciding what ERP-system to use in a Taiwan based electronics company. Pandey & Litoriya proposed the use of AHP with fuzzy numbers in order to compare and choose the best agile process for a specific project. AHP could similarly be used in this master's thesis to develop an evaluation model for JSFs.

4 Method

This master's thesis was divided into four parts. To start, research was conducted to understand what criteria are important when it comes to evaluating client-side JSFs and how a model for evaluation could be structured. Then a first draft of the evaluation model was constructed. Afterwards the model was tested by developing the same application using three different client-side JSFs, which stood as a basis for evaluating the JSFs regarding the found important criteria. Lastly, the evaluation model for deciding what JSF is best suited for a specific project was revised based on insights from the testing of the model.

4.1 Pre-study

In the first phase of the case study, a pre-study was conducted. This pre-study contained five parts, a literature search, interviews with employees at Exsitec, coding of criteria, a survey sent out to employees at Exsitec, and, lastly, deciding what evaluation criteria to evaluate the JSFs based on. These five steps are presented more in detail below.

After the pre-study was finished, the results were sent to the employees at Exsitec who participated in the interviews. This was done to gain potential feedback and improve validity as described by Runesson and Höst [29].

4.1.1 Literature search

In the first phase of the pre-study, a literature search was conducted to find important evaluation criteria regarding the adoption of a JSF. The focus during the literature search was to find important reasons for choosing and adopting a specific JSF, which can be seen as important criteria to evaluate the frameworks on. In addition, research into how one could construct a model for evaluating JSFs was also done. UniSearch[1], the Linköping University library search tool, was used to search for relevant peer reviewed studies. Search terms that were used in different combinations were "JavaScript", "Frontend", "Front-end", "Front", "End, "Frameworks", "Evaluation", "Comparison", "Criteria", "Factors", "Client", "Client-side", "Side".

[1] https://liu.se/biblioteket

After making a search, we read the titles of the articles that was given as a search result until articles with totally irrelevant titles started showing up. The abstracts of articles with the most relevant titles were read to get a further understanding of whether the article would be relevant for this thesis. If the article was potentially relevant the whole article was read and summarized as a first step. Sources cited by found articles also went through the same process if they seemed to be relevant to this master's thesis. Some of them were not peer-reviewed but acted as a complement to other peer-reviewed articles.

4.1.2 Interviews

Interviews were conducted with five employees at Exsitec to understand what they think are important factors when choosing a JSF. The interviews were semi-structured interviews as defined by Runesson and Höst [29], meaning that the questions were planned in advance but did not have to be asked in the planned order. Rather the conversation decided what questions were asked, with a mix of open and closed questions. This kind of interview gives more flexibility, enabling exploration and improvisation, compared to fully structured interviews with closed questions where the interviewer tries to find relations between constructs. The semi-structured interviews were held with employees at Exsitec who have roles where they have been/are working with front-end frameworks and where they have been in the position to discuss what client-side framework to choose in a project. The same questions were asked to all interviewees and the interview guide can be found in Appendix A. Firstly, the interviewees were asked about their previous experience in working with frontend frameworks. This was done to be able to exclude interviews from the analyzed data if the interviewees' experience were deemed insufficient. Secondly, they were asked questions about how projects at Exsitec are conducted and what criteria they think are important to consider when evaluating a JSF. At the end of the interviews the interview notes were recapped to the interviewees to avoid misunderstandings as stated by Runesson and Höst [29]. All of the interviewees gave consent to record the interviews, to enable later analysis of the interviews. After the five interviews were conducted further interviews were discussed. However, following the theory of theoretical saturation[11] and the fact that it was difficult to find interviewees it was decided to not continue since the last two interviews did not give any new criteria.

4.1.3 Coding

The results from the literature search and the interviews were analyzed through coding. Coding is described by Runesson and Höst [29] as the process of going through the qualitative data and giving parts of the text certain codes representing for example a theme. In this master's thesis the answers from the interviews and literature search were coded into different evaluation criteria, much like the methodologies used by Pano et al. [24] and Ferreira et al. [8].

The coding process started with analyzing the transcribed interviews one by one and preliminary codes were given to identify different evaluation criteria that were mentioned by the different interviewees. Each criterion that had not been mentioned before was given a new code. A second read-through of each transcription was done to further specify and refine evaluation criteria codes. Then, all criteria found when coding the transcribed interviews were summarized. After the coding of the interviews was done, the results from literature search were coded. Firstly, the criteria from different articles were coded. Secondly, these criteria were coded again to try to match criteria mentioned in the literature to criteria mentioned in interviews or coding it into a new criterion. This resulted in a summarized list of criteria mentioned in the literature and/or interviews.

4.1.4 Survey

The list of evaluation criteria that was derived from coding was used as a basis for the survey. The survey was sent out to 46 employees at Exsitec who work within application development. The survey inquired respondents about their experience in working with JSFs, their experience in decision making regarding the adoption of JSFs, and what their perceived most important factors regarding evaluation of JSFs are. The survey consisted of closed questions [26] in order to get quantitative answers to complement the qualitative answers from the interviews and the criteria found in the literature search. According to Phillips et al. [26], closed questions such as multiple choice or yes/no questions are categorical which allow for capturing of quantitative data on which categories the employees at Exsitec think are important when selecting a JSF. This as opposed to how Phillips et al. describe open-ended questions as good for exploration and qualitative analysis. The survey can be found in Appendix B.

4.1.5 Decision evaluation criteria

The result from the coding of interviews and the literature search was analyzed together with the results from the survey to achieve triangulation. Methodical triangulation is described as studying something using multiple methods [5], for example using a literature search together with interviews and a survey to research the same phenomenon. Triangulation increases the validity [29] and the credibility [5] if the results are convergent between methods. Results from the previous parts in the pre-study led to the decision of the most important evaluation criteria regarding adoption of a JSF.

4.2 Evaluation model first draft

The first draft of the evaluation model was developed to include the decided evaluation criteria. Inspiration for the model was taken from the studies conducted by Kaluza et al. [17], Thesing et al. [33] and Molin [20], who all developed models to help in the decision of adoption. Thesing et al. [33] study the adoption of a project management approach, whereas Kaluza et al. [17] and Molin [20] research the adoption of JSFs, as this master's thesis does. In addition, an implementation part was added to ensure that the user of the model got an understanding of the frameworks when evaluating it using the presented model. Thesing et al. [33] discusses the importance of comprehensibility when developing a model, meaning that the model should be easy and clear to follow. This was taken under consideration when developing the evaluation model. This first draft stood as a basis for the evaluation of the three frameworks and was intended to be updated after the implementation and evaluation with insights drawn from those parts of this master's thesis.

4.3 Testing of the Evaluation model

The first draft of the evaluation model was used to evaluate the three frameworks React, Angular and Vue. Firstly, applications in the three frameworks were implemented to gain an understanding of how the frameworks worked as well as to gain an understanding of necessary functionality for a working single page application. Secondly, the frameworks were evaluated based on the important criteria. Lastly, the results were summarized and analyzed. Testing the model was done to gain insights regarding how the model could be updated and improved.

4.3.1 Implementation

Implementing and developing with the frameworks was considered important to acquire a better understanding and help with the evaluation of the frameworks. On top of this, the

implementation part gave insights of what necessary functionality should be implemented when using the evaluation model. Three web client applications were developed using the three JSFs, React, Angular, and Vue. The three applications were connected to an already existing server developed previously by developers at Exsitec. The clients were developed in parallel so that one feature was developed in every framework before moving on to the next feature. For each feature, the order of development with the frameworks was varied. That is, if the first feature was developed using React first, then Angular and lastly Vue, the next feature would be developed in a new order, Angular, Vue and lastly React. Both of these measures counteracted getting more comfortable with one framework early on, which can lead to bias for that framework.

The implemented applications were starter applications, i.e., already constructed applications that developers at Exsitec can copy and build upon to reduce start up time in new projects. This was chosen as the task to be implemented since this master's thesis is done in collaboration with Exsitec. The basis for what would be developed in these starter applications were derived from an existing starter application developed in Angular, which is outdated. With this application in mind, a requirements specification was developed. When deriving the requirements, consideration was taken to try to include all basic parts that an SPA should include. The requirements were divided into different issues with underlying tasks which made up the project specification of the implementation. Each task was created once for every framework so there were three identical tasks, one for each framework. During the implementation one issue at a time was developed. The project specification can be found in Appendix C.

4.3.2 Evaluation

Evaluation of the frameworks was done in accordance to the evaluation model after the implementation was done. Inspiration was taken from Kaluza et al. [17] who evaluated the JSFs based on one criterion at a time. The evaluation was thereby performed in parallel on the three JSFs (React, Angular and Vue), evaluating one criterion on all JSFs and then moving to the next one, to ensure that the evaluation was performed as identically as possible. During this evaluation, notes were taken about important insights regarding the evaluation criteria and how these can be evaluated. This was done to be able to update and refine the evaluation model.

4.4 Evaluation model refinement

When the testing of the evaluation model was done the evaluation model was revisited and updated in accordance to insights gained during testing. This was done to improve understanding of the model and potentially improve transferability of the model to other kinds of frameworks than JSFs.

4.4.1 Review by Exsitec developers

After the refinement, the evaluation model was sent back to the developers who participated in interviews. This to give them the opportunity to review the model and give feedback regarding the criteria in the model and the design of the model. This was done to improve the validity of the model.

5 Results

This chapter presents the result from this master's thesis. First, results from the pre-study are presented in section 5.1 followed by the result of the first draft of the evaluation model. Then the results of the testing of the evaluation model is presented in section 5.3.1. Lastly, the revised evaluation model is presented in section 5.4.

5.1 Pre-study

This section presents the reader with result from the pre-sudy phase of this study. Result from literature search, interviews, survey and the decision of evaluation criteria are presented.

5.1.1 Literature search

The literature search gave insights of earlier research about evaluating JSFs. Out of all articles that were found during the literature search, three studies were used to understand important criteria, these being Ferreira et al. [8], Pano et al. [24] and Molin [20].

The important criteria from these articles are presented in chapter 3, both with descriptions and summarized tables. Table 3.1 presents the important criteria that Pano et al. found, table 3.2 presents the important criteria that Ferreira et al. found and table 3.3 presents the important criteria that Molin found. Research was also conducted regarding different evaluation models.

5.1.2 Interviews

Five interviews were held with employees at Exsitec. The interviewees were asked about their view on important criteria when evaluating a front-end framework and their general experience when it comes to front-end frameworks. All of the interviewees had multiple years of professional experience in working with front-end development, either through Exsitec or other companies. None of the interviewees were deemed to have insufficient experience. Two of the interviewees had been a part of Exsitecs decision to adopt Angular at the company, the other three had not been a part of any major decisions regarding front-end frameworks. The main focus of the interviews was to find what criteria the interviewees found important when

evaluating front-end frameworks, and they were asked to list everything they could think of that was important to them. The results from the interview period were five recorded and then transcribed interviews that later on were analyzed and coded.

5.1.3 Coding

The results from the literature search and interviews with employees at Exsitec were analyzed to form an overview of important evaluation criteria for a JSF. The transcribed interviews were analyzed for criteria, and these were coded using themes as described in section 4.1. The results from the coding of interviews are presented in table 5.1.

Table 5.1: Criteria mentioned in the semi-structured interviews with employees at Exsitec

Criterion	A	B	C	D	E	No. of occurrences
Easy to learn	X	X	X	X	X	5
Internal expertise	X	X	X	X	X	5
Maintained		X	X	X	X	4
Documentation	X		X	X		3
Suitability for the task	X	X		X		3
Third party support	X	X	X			3
Community	X			X		2
A clear structure			X		X	2
Short development time		X			X	2
Easy to develop with		X			X	2
External expertise			X			1

The important criteria found in the literature search were then analyzed and given codes in order to match criteria from different sources to each other. This resulted in a list of coded important criteria from the literature that are presented in table 5.2.

Table 5.2: Combined list of important criteria for adoption of a JSFs from the literature search

Criterion	Pano et al	Ferreria et al	Molin	No. of occurrences
Easy to learn	X	X	X	3
Maintained	X	X	X	3
A clear structure	X	X	X	3
Easy to develop with	X	X	X	3
Performance	X	X	X	3
Documentation	X	X		2
Community	X	X		2
Popularity		X	X	2
Internal expertise		X		1
Suitability for the task	X			1
Third party support	X			1
External expertise	X			1
Short development time	X			1
Size of code	X			1
To gain experience		X		1
Testability			X	1
Security			X	1
Maturity			X	1

Finally the criteria in table 5.2 were coded to match with criteria mentioned in interviews, see table 5.1. Each criterion was analyzed and given the code of one or multiple of the criteria found in the interviews, if possible. If not, it was considered a separate criterion. Most

criteria from the literature and the interviews overlapped, every criterion mentioned in the interviews were mentioned in the literature but some criteria from the literature was not mentioned in interviews. These were performance, size of code, to gain experience and testability. They were all coded into their own criterion. Popularity and maturity were not literally mentioned by interviewees but these two were described very similar in the literature and were coded to be a combination of maintained, documentation, community, third party support and external expertise. Security was also not literally mentioned in the interviews but was considered a sub-criterion of maintained since known security issues should be fixed by the framework maintainers. The result of the final coding was the list of criteria in table 5.3

Table 5.3: Results from final coding

Criterion	**Definition**
Easy to learn	The framework is easy to learn
Suitability for the task	The framework is suitable for the task, the important building blocks exist
Architecture	The framework has a clear structure for building applications
Usability	The framework is easy to use/develop with
Maintained	The framework is maintained and continuously updated
Documentation	The framework is well documented
Community	The framework has support from a big community
Internal expertise	Me/my team has previous experience with the framework
Third party support	The framework has many third-party libraries
External expertise	The framework is well known and many people have experience in using it
Short development time	The framework is quick to develop with
Performance	The framework produces applications with a high performance
Testability	The framework allows for testing of code according to industry standards
Get experience	To gain experience in using the framework
Size of code	The framework produces applications that are written in a small amount of code

5.1.4 Survey

The coded criteria in table 5.3 was the basis for the survey which was sent out to 46 employees at Exsitec. Out of the 46 developers who were sent the survey, 20 answered the survey. The respondents were asked about their view on important criteria. The survey had one question where the interviewed had to choose up to 8 criteria that they thought were important out of a combined list from literature search and interviews. They also had the opportunity to write their own criteria if they thought something was missing from the alternatives. Out of the 20 respondents, no one answered an own criteria, all chose among the presented alternatives. When asked if they ever had been involved in the decision of a front-end framework in a project, 7 out of 20 answered that they had and 13 out of 20 answered that they had not. The results of the evaluation criteria from the survey are presented in table 5.4. The criteria are presented as well as how many of the respondents that answered that the specific criterion was important in the evaluation of a front-end framework.

Table 5.4: Results from survey about important criteria.

Criterion	No. of occurrences
Documentation	19
Usability	14
Maintained	14
Suitability	14
Learnability	12
Community	12
Internal expertise	10
Architecture	9
Third party support	7
External expertise	6
Short development time	5
Performance	4
Testability	4
Get experience	2
Size of code	1

5.1.4.1 Chosen evaluation criteria

Based on the results from the literature search, interviews and survey a list of evaluation criteria which was to be evaluated in the evaluation model was derived. These evaluation criteria and their definitions are presented in table 5.5.

Table 5.5: Chosen criteria

Criterion	Description
Easy to learn	The framework is easy to learn
Suitability for the task	The framework is suitable for the task, the important building blocks exist
Architecture	The framework has a clear structure for building applications
Usability	The framework is easy to use/develop with
Maintained	The framework is maintained and continuously updated
Documentation	The framework is well documented
Community	The framework has support from a big community
Internal expertise	Me/my team has previous experience with the framework

The criteria in table 5.5 were chosen since they were featured often in the literature and interviews as well as being the eight most frequent answers from the survey.

5.2 Evaluation model first draft

The first draft of the evaluation model was developed based on the evaluation criteria found important in the pre-study of this master's thesis, the chosen criteria in table 5.5. These criteria and their definitions are the ones that are included in the evaluation model. The model includes four steps and these are presented below:

1. **Implement the task**

   Implement the task with the criteria in mind. Take notes on thoughts after implementing each issue in the specification. If multiple frameworks are planned to be evaluated using the model each item in the specification is recommended to be developed in each framework before moving on to the next item. It is also recommended to vary the order in which the frameworks are used. Both of these measure help to minimize a bias towards a certain framework due to learning it first.

2. **Evaluate the framework**

   Evaluate the framework based on every criterion, one at a time. Take notes and score the framework based on each criterion using the following scale:

   - (4) The definition of the criterion is fully applicable
   - (3) The definition of the criterion applies to a large extent
   - (2) The definition of the criterion is partially applicable
   - (1) The definition of the criterion applies only to a very limited extent
   - (0) The definition of the criterion does not apply

   The criteria that should be evaluated is presented in table 5.6 below. A description for each criterion is presented to give the user a better understanding of what the criterion covers.

Table 5.6: Evaluation criteria

Evaluation criterion	**Description**
Easy to learn	The framework is easy to learn
Suitability for the task	The framework is suitable for the task, the important building blocks exist
Architecture	The framework has a clear structure for building applications
Usability	The framework is easy to use/develop with
Maintained	The framework is maintained and continuously updated
Documentation	The framework is well documented
Community	The framework has support from a big community
Internal expertise	Me/my team has previous experience with the framework

3. **Calculate the result**

   Calculate the total score from the evaluation of the framework, summarize the scores for all criteria.

4. **Analyze the result**

   Analyze the total score combined with the qualitative insights gained during the implementation.

The final evaluation model included an instruction for an implementation phase which will ensure that the user of the model gets an understanding of relevant concepts regarding SPA development in the evaluated framework. In this first draft of the model the basis for implementation was the project specification, see appendix C, developed in collaboration with Exsitec which included implementing many relevant SPA concepts. When finalizing the evaluation model the project specification was simplified into a smaller task that is faster to implement.

5.3 Testing of the evaluation model

In this section the results of testing the evaluation model are presented. First results from the implementation of starter applications for Exsitec using the JSF's React, Angular and Vue are presented. The implementation part is followed by the evaluation of the mentioned frameworks, using the first draft of the evaluation model.

5.3.1 Implementation

This section presents the results from the implementation phase of the master's thesis where every subsection corresponds to one implemented feature in the applications. The order in which they are presented is the same as in which they were developed. The order in which the frameworks were used was based on a rolling schedule which was randomized on the first feature. The randomized order was React, Angular and then Vue, shifted one step left on every new issue. The application that was developed with the three frameworks was a starter application for new projects that could be used by teams at Exsitec. The reason for choosing this as the implemented task was that this master's thesis was done in collaboration with Exsitec and that it could be useful for employees at Exsitec to have this. The implementation of the three web applications took approximately six weeks.

5.3.1.1 Setup application

This feature included setting up a new application that could be started on localhost and viewed through a web browser. The application in question was a simple Hello World app[1] that displayed the text "Hello, world!" to the user.

Setting up the Angular and Vue applications was handled with the respective framework's command line interfaces (CLI). However, for React there does not exist an official CLI but in the React documentation it is recommended to use the create-react-app[2] tool to setup your application.

5.3.1.2 Setup Routing

During this feature four different components were added. These were a Home page which displayed the string "home page" and was accessed through the "/home" route. An Examples page that displayed "examples page" and was accessed through the "/examples" route. A component which was always visible and handled navigation between these two and was accessed through the "/" route. And lastly, a page which handled every other route and informed the user that there was nothing there.

In Angular creation of components and the setup of routing was handled with the Angular CLI which created components and handled integration of them into the app as well as creating a general routing setup with Angular's built in routing tool. In React no CLI tool exists and in Vue the CLI does not handle tasks like these. In React the React Router[3] tool was used and the tutorial on the tool's webpage was used. For Vue the Vue Router[4] tool was used and the documentation was followed to achieve the routing setup.

5.3.1.3 Authentication

Authentication was setup in each application with the help of auth0[5] which is a platform handling authentication and authorization for other applications. With auth0 login, registration, and logout functionality was added to each app, as well as displaying the email of the currently logged in user in the home component.

Very simple auth0 software development kits (SDKs) exist for React and Angular that are specific for each framework. Quickstarts [6] for these SDKs were followed to integrate auth0

[1] https://sv.wikipedia.org/wiki/Hello_World
[2] https://github.com/facebook/create-react-app#create-react-app--
[3] https://reactrouter.com/
[4] https://router.vuejs.org/
[5] https://auth0.com/
[6] https://auth0.com/docs/quickstarts

into the applications. For Vue, a general JavaScript SDK exist and a quickstart for this was followed, however the quickstart was not working for Vue 3. After a while of searching, a community thread was found where a person had developed sample code with auth0 JavaScript SPA SDK that worked with Vue3. This sample code[7] was used to integrate auth0 into the Vue application. Later on, a beta version of a Vue SDK for Auth0[8] was released, however it was not recommended for production mode.

5.3.1.4 State Handling

State handling was implemented through being able to input some text into a text field which is handled with local component state and then being able to save the input text which puts the text in a list saved in the global state. This was done in the examples component and the bits of text are called examples in the state.

The local states in all frameworks were implemented according to the framework's documentation. In React, useState [9] was used in order to keep track of the input in the text field, which enforces one-way data binding. This makes it necessary to write an additional function which handles updating the state when the user inputs text. In Angular NgModel[10] was used and in Vue v-model[11] was used, both of them leveraging two-way data binding, meaning that the state is automatically updated from the view on user input.

The global states were all handled with external libraries made specifically for state handling. In React, Redux[12] was used with Redux Toolkit[13] which simplifies development with Redux. In Angular, NgRx[14] was used which is a state handling tool specifically made for Angular which was inspired by Redux. In Vue, the plan was to implement Vuex[15] but when reading the Vue documentation on state management it was noticed that Vuex would be replaced by a new tool called Pinia[16]. Therefore, Pinia was used to handle the global states in Vue.

5.3.1.5 CRUD

CRUD (create, read, update, delete) was implemented through connecting the front-end applications to the backend server so that the examples handled in the state handling step could be saved in a database. The requests to the back-end were implemented in the state handling tools so that the global states were in sync with the data in the database. Four requests were implemented in each tool. The post request handles creation of examples. The get request handles fetching of examples created by the user that is currently logged in. The put request handles changing the text in the example. The delete request handles removal of examples.

Angular has a built-in tool for handling HTTP-requests, HTTP-client [17], that was used to communicate between the client and server in the Angular application. In Angular, the connection between the state and HTTP-requests was implemented with NgRx effects [18] which handles actions connected to external interactions.

[7] https://github.com/lstyles/vue3-auth0-sample/
[8] https://auth0.com/docs/quickstart/spa/vuejs-beta
[9] https://reactjs.org/docs/hooks-state.html
[10] https://angular.io/api/forms/NgModel
[11] https://vuejs.org/guide/essentials/forms.html
[12] https://redux.js.org/
[13] https://redux-toolkit.js.org/
[14] https://ngrx.io/
[15] https://vuex.vuejs.org/
[16] https://pinia.vuejs.org/
[17] https://angular.io/api/common/http/HttpClient
[18] https://ngrx.io/guide/effects

Vue and React do not have built in tools for handling HTTP-requests, meaning that a third party package had to be used. For handling HTTP-requests in the Vue and React applications, Axios [19] was used. In Vue, the connection between HTTP-requests and state was implemented by using async/await syntax in the action and making HTTP-requests directly in the action. In React, the connection was done by using CreateAsyncThunk in Redux toolkit [20] to create async actions to update the state with the response from the server.

5.3.1.6 Authorization and protected routes

In this part of the implementation protection of endpoints in the server was added resulting in the client having to send tokens with the HTTP-requests to be able to authorize themselves and communicate with the server. This was implemented with the Auth0 SDKs in the clients. All routes were protected using Auth0 so that only authenticated users can view the pages. In addition, a new admin route was added and protected with custom made route guarding in all the frameworks. The admin route guards made use of Auth0 tokens to check if the logged in user was an admin or not.

In Vue and React, the authorization was implemented by getting the logged in user's token using the Auth0 SDK and sending this token in the headers with every request to the server. In Angular, Auth0 has implemented an HTTP-request interceptor that intercepts the request to a specific URI and adds the token to the request automatically.

In Angular and Vue, the routes were protected using built in route guards in Auth0. In the React application, a custom protected route for the React Router was implemented using a function in the Auth0 SDK which checks for a logged in user and a custom function for handling redirects from Auth0. The admin route guards were all custom made with the framework's respective routing tools and tokens from Auth0.

5.3.1.7 Design

The design of the applications was implemented using Material Design[21] libraries for the different frameworks. The design of the applications was implemented to be as similar as possible in the different frameworks.

In every framework the main tutorial was used for setting up the design library and then the same design process was conducted for every application, starting with the layout of the app and then moving on to implementing the design of the different views. In Angular the library Angular Material was used[22], in React the library MUI[23] was used and in Vue the library Vuetify[24] was used.

5.3.2 Evaluation

This section presents the result of the evaluation of the frameworks using the first draft of the evaluation model. The criteria are evaluated one by one on all three frameworks. It is important to note that this is the subjective thoughts of the authors of this master's thesis and should not be taken as concrete facts. The model is intended to give the evaluator a chance to evaluate the framework based on their thoughts.

[19] https://axios-http.com/docs/intro
[20] https://redux-toolkit.js.org/api/createAsyncThunk
[21] https://material.io/design
[22] https://material.angular.io/
[23] https://mui.com/
[24] https://next.vuetifyjs.com/en/

The criterion "Suitability for the task" refers to the framework being suitable to use when developing the intended product. Therefore, when testing the model on React, Angular and Vue in the following sections, the "Intended product" will be the starter application that was implemented in section 5.3.1.

The criterion "Internal expertise" is evaluated by checking if the intended developer/team have previous experience with using the framework. This is not relevant to the testing of the model since no intended team exist. Therefore, this criterion is excluded from the testing of the model but should be included when using the final model. However, the authors of this thesis have previous experience in React and Angular. In order to give a fair evaluation, measures has been taken to minimize the impact of this previous knowledge during the both the implementation and evaluation parts of the model testing.

5.3.2.1 Evaluation of React using the evaluation model

The first framework that was evaluated was React. In the following section the four steps of the evaluation model are followed and presented.

Step 1 - Implement the task

The implementation of the task using React has already been conducted. The results from the implementation can be found in section 5.3.1.

Step 2 - Evaluate the framework

In the following paragraphs the React framework is evaluated for all criteria. Each criterion is discussed and a score is given.

Easy to learn - **Score: 3**

There are both positive and negative thoughts regarding learnability when using React.

To have all functionality for one component in the same file (logic, UI and (if wanted) styling as well) made it easy to learn the framework and start producing code since all logic regarding one component is gathered in one place. On top of this, it was easy to find good tutorials using the React community and therefore making it easier to learn the framework.

The negatives are that it is hard to understand how to make a new component. Since a React component only is a function that returns a JavaScript XML element you can structure it however you want and store it anywhere you want. This can be confusing if compared to other tools using advanced CLI's that can provide a boilerplate for new components with a simple command. The documentation of React also makes it a bit hard to learn React the way it is used in modern applications. This since much of the documentation concerns class based components which is an outdated way to use React.

Overall, React was quite easy to learn.

Suitability for the task - **Score: 4**

For the implemented application React was deemed suitable due to it being a relatively simple framework and since the task was relatively small. There were no parts of the task that felt like it was not suited to use React as a front-end framework, and all important building blocks exist. Furthermore, React has a good third party support, it was for example very easy to use the Auth0 SDK for React applications.

In the future if the starter application is the base for a large project with many developers React could be extended to introduce type checking and with a clear rule for developers one could enforce a certain structure of the code base. This would allow React to be used efficiently in a large project.

Architecture - **Score: 2**

React uses a component based architecture where UI elements are built up by different components. A component itself can also be built up by smaller components. However, when using React there are no specific requirements to follow regarding how to structure these components which could potentially lead to unclear code.

When it comes to file structure there are no decided way of structuring the project, which forces the developer to take the decision of how to structure the project. However, there are best practices in the React community and also recommendations in React's documentation that can be followed if the developer chooses.

Usability - **Score: 4**

After implementing the task using React the opinion is that React is easy to use and develop with. Creating working components goes quite quickly and without any major difficulties. The framework is quite straightforward to use and after getting used to it the development flows smoothly. The project does not include excessively many files. One drawback is the lack of two-way data-binding which presents the need for a lot of small functions updating local states in the components.

Maintained - **Score: 4**

React is a very popular framework that is continuously updated. The latest update, as of the time this master's thesis is written, was released on March 29th 2022. During implementation we never got into any issues regarding incompatible third party frameworks.

Documentation - **Score: 2**

Following the React documentation was at times misleading. For example, when developing local state handling the documentation recommended implementing class based components when the modern way to use React is by implementing functional components. The same is true for the tutorial provided on Reacts website. The authors of this master's thesis had developed using React before, resulting in previous knowledge about the recommendation to use functional components. However, if the authors would not had this knowledge from earlier, the application would probably have been implemented in class based components.

Since many functionalities that are needed to build a SPA is not included in React, such as routing and an HTTP client, documentation for these parts does not exists. This results in having to rely on tutorials or the React community to understand which tools to use for a specific purpose, which sometimes leads to uncertainty about if the correct/best tool is used. Many times these tools have their own documentation to follow. However, this is more related to the community criterion rather than documentation.

Community - **Score: 4**

During implementation, React was deemed to have a big and good community. Many issues that were encountered were easily resolved by finding some other developer in the React community who had solved a similar problem before. One example being when implementing protected routes with Auth0 and React Router v6. The Auth0 documentation was made

with React Router v5 but it was solved thanks to the React community having encountered the issue before.

Step 3 - Calculate the result

The summarized score for React is calculated to 23 points out of 28 possible.

Step 4 - Analyze the result

Overall React is deemed to be a framework that is easy to learn and use which has a big community which is helpful when developing. It is also continuously maintained. For the task at hand, it is deemed to be suitable. However, some drawbacks exists. The documentation is at times a bit outdated which can cause issues when trying to learn React. Although, this is counteracted by the helpful community where many good tutorials exist. Another drawback is the free structuring of React which sometimes can make it confusing knowing how to structure the application.

5.3.2.2 Evaluation of Angular using the evaluation model

The second framework that was evaluated was Angular. In the following section the four steps of the evaluation model are followed and presented.

Step 1 - Implement the task

The implementation of the task using Angular has already been conducted. The results from the implementation can be found in section 5.3.1.

Step 2 - Evaluate the framework

In the following paragraphs the Angular framework is evaluated for all criteria. Each criterion is discussed and a score is given.

Easy to learn - **Score: 2**

The Angular CLI can be used to create the shell for components and modules making it quite easy to learn how components and modules should look like. There is also good documentation on the Angular web page making it easy to learn to use the framework. However, since there are four files for every component the project grows fast making it hard to get a good overlook over the complete project. This is something that makes it harder to learn the framework, especially if user does not start from scratch and instead use the CLI to create components and modules. Observables[25] is used in state handling and is quite hard to understand making it harder to learn the framework. On top of this, TypeScript is more complicated to use than JavaScript which can be seen as Angular is harder to learn.

Suitability for the task - **Score: 4**

Implementing the task using Angular shows that Angular is suitable for building a single page application, all the important building blocks exist. Angular has a lot of built in functionalities that other JSFs do not have, making it easy to solve the task and knowing that the correct tools are used. On the other hand, for a simple task Angular can be seen as quite messy to use, since a lot of boilerplate code and files are created. However, if the project should grow it would be even more suitable to use Angular which enforces a good structure and typing.

[25] `https://angular.io/guide/observables`

Architecture - **Score: 3**

Angular has a clear component based architecture. The components themselves can be further separated into different modules which separate different parts of the application. This achieves a clear separation of concerns. The Angular CLI is also very helpful in creating the architecture which makes sure the application is structured in a good way. One drawback is that sometimes it is unclear what should be a module and what should be a component.

Usability - **Score: 3**

The Angular CLI is a good tool which facilitates the development. Angular also has a lot of tools included which facilitates decision making regarding what tools to use - you use Angular's tool. The requirement to assign a type to variables in TypeScript means that the code takes longer to write. However, this facilitates debugging and development making it more easy to use and to produce functioning code. It can at times be hard to gain a high level understanding of an Angular project if you are not familiar with the project or very comfortable and used to working with Angular. The concept of Observables is also a bit hard to understand and seems unnecessarily difficult.

Maintained - **Score: 4**

Angular is continuously updated and the latest update was released on the November 4th 2021. Angular is also backed by Google which can be seen as reassuring. During development there were never any issues regarding incompatible third-party frameworks.

Documentation - **Score: 3**

Angular's documentation is well written and contains updated information. However, sometimes the documentation is quite hard to follow since it always requires two files to understand the logic and see the functionality. For example, if the developer should read the documentation about form inputs, it has to read documentation including instructions for both the HTML-page as well as the typescript-file.

Community - **Score: 3**

It is easy to find help from the Angular community since it is a big framework which is used by many other developers. One issue is that, like in Angular's documentation, most problems involves both the HTML file and typescript file which makes a bit harder to read. Another issue is that at times the help found in the community can be a couple of years old. However, Angular has been stable for a long time and the solutions that can be found often works even though they are a couple of years old.

Step 3 - Calculate the result

The summarized score for Angular is calculated to 22 points out of 28 possible.

Step 4 - Analyze the result

Overall, Angular is a framework which produces well-structured applications and facilitates development with the use of TypeScript and a good CLI tool. Angular has a good community, is well documented and continuously maintained. Some drawbacks of Angular are that it can be seen as challenging to learn and at times difficult to use. It can also be hard to gain a high level understanding of an application if you are not familiar with the project.

5.3.2.3 Evaluation of Vue using the evaluation model

The third framework that was evaluated was Vue. In the following section the four steps of the evaluation model are followed and presented.

Step 1 - Implement the task

The implementation of the task using Vue has already been conducted. The results from the implementation can be found in section 5.3.1.

Step 2 - Evaluate the framework

In the following paragraphs the Vue framework is evaluated for all criteria. Each criteria is discussed and a score is given.

Easy to learn - **Score: 4**

After developing the task using Vue, the conclusion is that Vue is an easy framework to learn. There are not a lot of code that is necessary to create components. Another thing that makes Vue a framework that is easy to learn and understand is that it has a very clear division of the code connected to a component. The component is divided into template, script and style but the three parts are still in the same file.

One thing that made it harder to learn was that during the implementation-part of the task using this framework, Vue was in the transition from Vue2 to Vue3. This resulted in that it was a bit difficult to follow the documentation that often primarily showed instructions for Vue2. Help from the community was also primarily centered around Vue2. However, this was a coincidence and should not be a problem in the long run.

Suitability for the task - **Score: 3**

Vue is overall seen as a framework that is suitable for the task. All necessary building blocks exist and it was easy and went fast to develop the task. Vue is a relatively new framework, especially with the new Vue3 version, resulting in that it does not exist many well working third party integration's for Vue as many other frameworks. For example, when integrating Auth0 into the application, there was better and easier SDKs to follow for React and Angular than for Vue. However, Vue is growing and is getting more and more popular.

Architecture - **Score: 3**

Vue is a component based framework which has a clear separation of code resulting in a quite easy architecture to follow.

When it comes to file structure there are no decided way of structuring the project, which forces the developer to take that decision. However, there are best practices in the Vue community and also recommendations in Vue's documentation that can be followed if the developer chooses.

Usability - **Score: 3**

After developing using Vue the impression is that it is a framework that is easy to use due to not needing much code to create functionality. Because of this, the development goes quickly. Vue has two-way data-binding, something that simplifies handling local states. However, one drawback was that the transition from Vue 2 to Vue 3 at times complicated development. This was not taken into much consideration n the evaluation since this is only temporary.

Maintained - **Score: 4**

Vue is a framework that is continuously updated, the latest version was released February 12th 2022, as of the time this master's thesis is written. On top of this, it is a framework with a growing popularity. If one is indulgent with the fact that the task was implemented in the middle of the change between versions, there are no major difficulties with third party integrations.

Documentation - **Score: 3**

The documentation is well written and easy to follow. There have been some difficulties when following the documentation since you end up in the documentation for Vue version 2 when searching for Vue documentation. However, this is mainly since the implementation took place in the middle of a change between versions. As of writing this the main documentation is changed to have Vue 3 as default.

Community - **Score: 3**

The experience of the Vue community is good. A lot of help has been taken regarding the difficulties with the change from Vue 2 to Vue 3. One example of this is the problems with the Auth0 JavaScript SDK not working with Vue 3. Here help was taken from the Vue community where working solutions for Vue 3 had been created by other developers. This shows that the Vue community is active and a good help to web developers.

Step 3 - Calculate the result

The summarized score for Vue is calculated to 23 points out of 28 possible.

Step 4 - Analyze the result

After evaluating Vue using the evaluation model it is concluded that Vue is an easy framework to learn and that it is easy to use. Within the Vue components the code has a clear separation but the file structure of the app is up to the user which can lead to confusion. Some difficulties using the frameworks existed, however these were mostly connected to the transition from Vue 2 to Vue 3. The community and documentation were very helpful and Vue is a framework that is continuously updated.

5.4 Evaluation model refinement

This section presents insights gained during the testing of the implementation model as well as the refined evaluation model.

5.4.1 Insights on refinement

One of the reasons for testing the evaluation model was to get insights about how the model can be improved. During the use of the evaluation model insights were documented and these are presented below.

- When using the model, it was sometimes difficult to differ the criteria from each other resulting in parts of evaluation that were included in multiple criteria evaluations. Therefore, one improvement of the model was to update some criteria. This was done to try to make the model easier to follow and hopefully give a more clear result.
- From the testing of the model it was determined that it would be beneficial to clarify what the different criteria mean. Due to this definitions and names of some criteria

were changed to better reflect what the criteria mean. In addition, a description part was added to each criterion to further clarify the criterion and at times give examples about what is meant by a criterion.

- The scoring seemed a bit unnecessary when using the model. All the evaluated frameworks got almost the same score and it was difficult to analyze the scores. However, the scoring was kept in the refined version of the evaluation model since it requires the evaluator to reflect about the framework regarding the criterion. Moreover, the evaluated frameworks in this master's thesis were quite similar but this is not necessarily the case for a future user of the model, resulting in that the scoring part of the model can be helpful for a decision.
- Insights about the implementation part of the refined model was also gained during testing. For the refined model the project specification was downsized in terms of issues and each issue was clarified. A prototype that shows the thought of layout of the app that the evaluator can follow was also added.

5.4.2 Refining the evaluation model

In this section the refinements of the model are presented. Three major refinements were made. An update of the criteria, clarification of the criteria as well as adding a thorough implementation guide.

5.4.2.1 Update of criteria

When evaluating the frameworks using the model the criterion "Easy to learn" seemed hard to evaluate on its own. It was evaluated as a combination of the "Usability", "Documentation", and "Community" criteria. Therefore, the criterion "Easy to learn" was removed and instead the descriptions of the mentioned criteria was updated to make sure that the qualities that makes the framework easy to learn will be taken into consideration by the evaluator.

5.4.2.2 Clarification of criteria

Updated definitions for the criteria seemed necessary for a clearer model to follow. In addition to that, a description was added to each criterion to facilitate the evaluation for the user of the model.

The definitions and descriptions for the criteria can be found in table 5.7.

Table 5.7: Criteria in evaluation model

Criterion	Definition	Guiding questions
Suitability for the task	The framework is suitable for the task	Do all necessary building blocks to build the planned app exist? Do you believe the task can be implemented using the framework in the expected time frame? Do necessary third part integrations exist and work for the framework?

Architecture	The framework has a clear structure for building applications	Does the framework have a clear structure for partitioning the application? Does the framework's way of structuring apps fit the way of working in the planned team/project?
Usability	The framework is easy to develop with	Does the framework have tools that aid in development? Do you think that the framework is easy to understand? Can you develop quickly using the framework?
Maintained	The framework is maintained and continuously updated	Is the framework supported, for example by a company or a large open source community? Is the framework improved regularly with new functionality? Does the framework regularly fix bugs and other issues?
Documentation	The framework is well documented	Is the documentation easy to follow? Is the documentation helpful when trying to learn the framework? Does the documentation cover all functionality of the framework? Is the documentation updated when new functionality is introduced?

Community	The framework has support from a big community	Is the framework popular and used by many? Does the community offer good help and support for the framework? Does the community offer helpful tutorials to aid in learning the framework?
Internal expertise	Me/my team has previous experience with the framework	What experience do the planned team members have with the framework? Is it important to use a a framework that the team knows or is it accepted to learn a new framework?

5.4.2.3 Implementation guide

To ensure that the evaluator has enough experience to use the evaluation model on a JSF a thorough implementation guide was added. This guide made sure the evaluator has gotten acquainted with important SPA concepts and how to implement them with the JSF. The suggested task that should be implemented using the framework that is to be evaluated is a TODO-list where items can be added, updated and removed. The specification for this implementation task can be found below in table 5.8. The prototype that displays how the application can be designed can be found below where figure 5.1 displays the welcome view and figure 5.2 displays the todo list view.

Table 5.8: Specification for implementation guide

Issue	**Description**	**Definition of done**
Setup application	Setup the application so that the application can be run locally and viewed through the browser.	The application can be viewed from the browser.
Routing	Implement routing and enable navigation between the two views, welcome screen and todo-list.	One can navigate between the welcome screen and todo-list views.

Todo-list	Implement functionality for a todo-list. New items should be able be added to a list on the todo-view. These items should be stored globally in the application so that the items remain when navigating between the views but do not have to remain when reloading the application. The first item in the todo-list should be presented on the welcome-screen.	The todo-list view should enable the user to add items to a list. The items in the list should remain when navigating between the views. The first item in the list should be displayed on the welcome screen.
CRUD	Enable the user to create, read, update and delete items in the todo-list by making http-requests to the provided server (GET, PUT, POST & DELETE). The changes on the server should be reflected by the front-end application.	The user can create an item in the list that is saved by the server. The user can update an item in the list that is updated in the server. The user can delete an item in the list that is deleted in the server. When reloading the application the items is fetched from the server. The changes on the server is reflected by the frontend application by handling the response from the server for each request.
Design	Implement design in the application based on the provided prototypes. Choosing how to implement the design is up to the user.	The application looks relatively similar to the provided prototype

5.4.3 Refined evaluation model

The refined evaluation model is presented below. Criteria was updated with descriptions to aid in evaluation and instructions for an implementation task is included. The refined evaluation model includes four steps, Implement the task, Evaluate the framework, Calculate the result, and Analyze the result. The final evaluation model can be found in Appendix D

1. **Implement the task**

   The task that should be implemented is a todo-list where the user should be able to create, update and delete items from the list. This task is only a suggestion and if the user of the model have another task that is more suitable, this can be used instead. The specification for the task can be found in table5.8.

   Implement the task with the criteria in mind. Take notes on thoughts after implementing each issue in the specification. If multiple frameworks are planned to be evaluated using the model each item in the specification is recommended to be developed in each

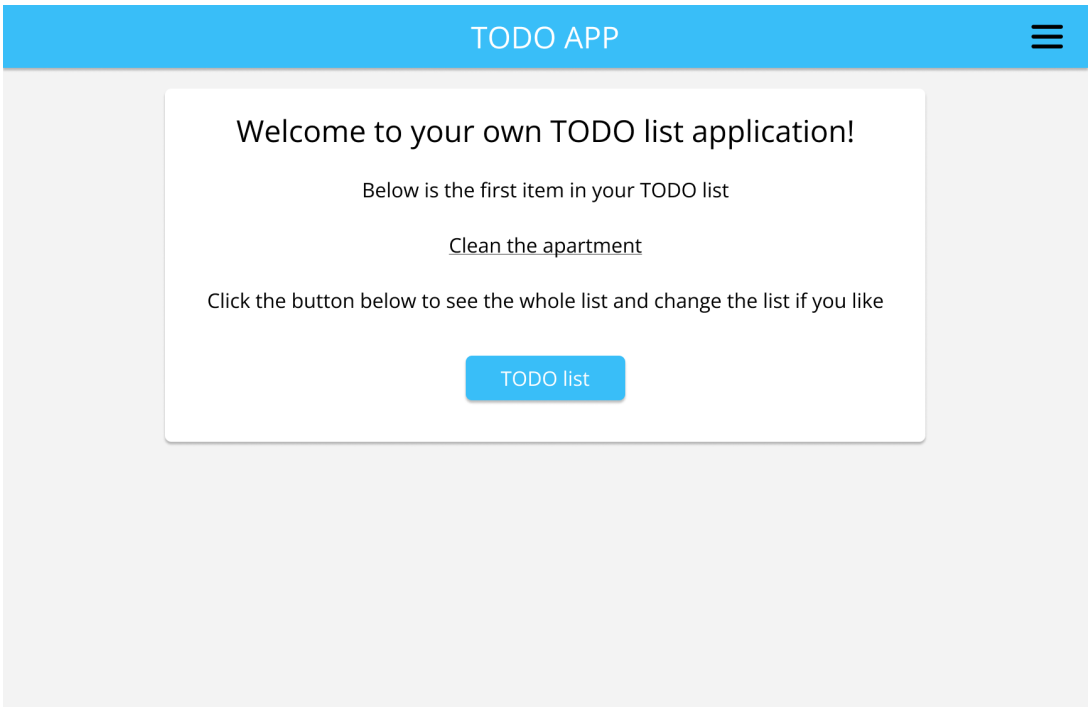

Figure 5.1: Welcome view of todo application prototype

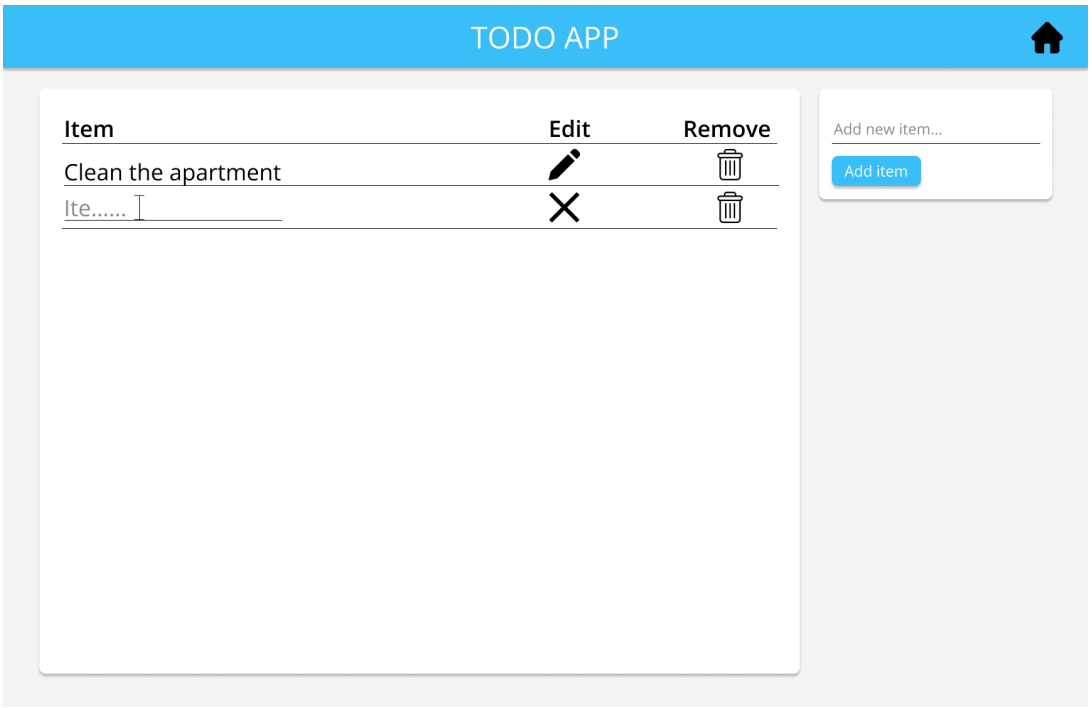

Figure 5.2: Todo list view of todo application prototype

framework before moving on to the next item. It is also recommended to vary the or-

der in which the frameworks are used. Both of these measure help to minimize a bias towards a certain framework due to learning it first.

The CRUD issue in the specification can be implemented with the help of this server application[26] that the authors of this thesis has developed. Read the README file in the project to get more information regarding how to use it.

2. **Evaluate the framework**

   Evaluate the framework based on every criterion, one at a time. Take notes and score the framework based on each criterion using the following scale:

   - (4) The definition of the criterion is fully applicable
   - (3) The definition of the criterion applies to a large extent
   - (2) The definition of the criterion is partially applicable
   - (1) The definition of the criterion applies only to a very limited extent
   - (0) The definition of the criterion does not apply

   The notes and scores should later be used to analyze the framework. For each criterion there are aiding questions that can be used as a basis for evaluating the framework for that criteria. These questions is only a suggestion and are presented to give an understanding of what the criterion covers. If the user of the model have own thoughts about what is important for a criterion other/more questions can be included as well. The criteria that should be evaluated is presented in table 5.7.

3. **Calculate the result**

   Calculate the total score from the evaluation of the framework, sum the scores for all criteria.

4. **Analyze the result**

   Analyze the total score combined with the qualitative insights gained during the implementation. Try to think about what evaluation criteria are important for the planned team/project.

5.4.4 Review by Exsitec developers

Out of the five interviewed developers at Exsitec one was able to take the time to review the evaluation model. The developer generally thought that the evaluation model looked good and that the model could fulfill its purpose of evaluating JSFs. However, the developer had some feedback.

One issue raised was if it was necessary to specify the implementation task as much as it is in the model. The developer thought that it might be better to let the user of the model decide by themselves what app to develop and give guidelines to what should be included. This was not changed in the model since the implementation task was created with this in mind. The specification of what should be implemented was based on what the authors of this thesis believes are the important things to understand when using a JSF. In the model it is also stated that the evaluator can do another implementation task if they find it necessary. However, the description of the first step of the model was changed to specify that the specification of the task was developed with the purpose to include what the authors of this thesis find to be important in SPAs.

[26] `https://github.com/ellensundholm/todo-backend`

The developer also brought up that the model could be used in a more general way to evaluate other types of front-end frameworks that are not JavaScript based.

Lastly, the developer brought up that a weighting system would be beneficial to the model. Weighting of criteria was not added to the model but the last step of the model was clarified to ensure that the user understands that reflection regarding how important different criteria are to them is a part of this step.

The last two of these issues will be discussed further in the discussion chapter of this thesis, see chapter 6. The final evaluation model is found in appendix D.

6 Discussion

This chapter contains a results discussion in section 6.1, a method discussion in section 6.2, and a discussion about this master's thesis in a wider context in section 6.3.

6.1 Results

In this section both the results regarding the evaluation criteria that was found to be important and the finished evaluation model will be discussed.

The criteria that employees at Exsitec thought was most important mostly corresponds with the results from the literature review. All criteria mentioned by Exsitec employees in interviews were mentioned in the literature but there were some variations regarding which criteria was mentioned the most. The fact that all answers coincided with the literature was expected since the results from literature review mainly consisted of insights other authors gained from interviews with developers.

One thing that stands out was that performance was mentioned by all of the articles used to gather criteria but was not mentioned in any of the interviews. Another stand out is that internal expertise is valued highly among Exsitec employees but was only mentioned in one of the articles. These two variations could be due to the company setting of Exsitec. Since it is a consultancy firm helping other companies with digitization, application development is not its main task. Therefore, performance might be pushed aside for other criteria that ensures a smooth working process for building apps. I.e., the focus is to develop applications quickly and smoothly, thus not seeing the need for applications to be as performant as possible. This argument could be made for why they also value internal expertise very highly. Another argument for internal expertise is that they develop and maintain multiple applications at once, making it very important that the framework chosen can be handled by many employees. Another company with a different setting might value the criteria differently.

Runesson and Höst [29] mentions that one way to increase the validity of a study is to send results back to participants in the study. Web developers at Exsitec had two opportunities to give feedback about the results in this master's thesis, first on the evaluation criteria list

derived when answering research question one and then on the refined evaluation model developed when answering research question two. The interviewees thought the criteria list was good and did not mention any missing criteria. Regarding the evaluation model, the interviewee that returned feedback thought the model was good and had no major input except some smaller issues/comments.

One aspect that was brought up by the developer was if all criteria should be equally important or if they should weigh differently when using the model to evaluate a framework. This was something that was discussed a lot during the development of the evaluation model. One idea was to include different weights based on what how many votes the criteria received in the survey. However, in the end the conclusion was that it was better that the users of the model themselves should reflect on what was important criteria for them and take this under consideration when summarizing the results for the framework. Criteria important for one individual or team may not be the same criteria that is important for others. Therefore, another idea was to include a step in the model which included that the user should weigh all criteria based on a set scale, like Thesing et al. [33] did in their model. However, this was also chosen not to be included in the model since Thesing et al. [33] highlighted the importance of a comprehensive and easy model and this step would make the model more complex. However, since the developer thought it was not clear enough that the evaluator themselves should reflect on which criteria they think is most important for their specific case, this part of the evaluation model was rewritten to be more clear.

Another aspect brought up by the developer at Exsitec was that the evaluation model might not be limited to evaluating JSFs and could be used to evaluate other front-end frameworks as well, and the authors of this master's thesis agrees. Since all sources used in the pre-study about important evaluation criteria focused on JSFs, one can not be sure that the evaluation model can be transferred to other front-end frameworks. However, there is a possibility that the model is suitable for other types of frameworks as well and it would be interesting to investigate this further. Therefore, to investigate the transferability of the evaluation model to other front-end frameworks was brought up as a recommendation to future work in 7.1. One potential problem could be that when trying to use the model to compare a JSF and a framework in some other language there might be technical aspects that are beyond the scope of the model developed in this thesis. This since all criteria in this evaluation model are non-technical. However, these criteria should be important even if the programming language is different.

The evaluation model includes a step where the user should implement a small task using the framework, with the aim of acquiring enough knowledge about the framework to be able to evaluate the framework based on the criteria. The recommendation was to develop a Todo-application based on derived specifications or if the user wants, develop an own project. Here, an addition could have been to recommend the user of the evaluation model to implement a sub-set of the project at hand if this is known. This would allow the user to better understand if the framework is suitable for the project.

6.2 Method

The method for deriving the evaluation model included a selection of the most important criteria that should exist in the model. This selection was done partly based on results from the interviews and survey, which was conducted with employees at Exsitec. A limitation to this method is that only employees from one company have been interviewed about their view on important criteria. The results from this has then been used as a basis when developing the model. The credibility of this study would have been improved if a bigger and more distributed sample of interview objects would have been used.

To improve the replicability of this master's thesis the description of the methodology used has been written to try to make it as precise and clear as possible. This to ensure that another person can understand what has been done and try to do it themselves.

One threat to replicability is that the codes used during the coding process was developed simultaneously as the coding, based on criteria mentioned in interviews. This is called inductive coding [18]. To improve replicability, one could have used deductive coding as stated by Linneberg and Korsgaard [18] instead. This would entail pre-developing codes based on the theory found during the literature search. If the codes had been developed earlier another researcher would have already developed codes to use, improving replicability. The approach used in this thesis could lead to small differences in codes. However, the results of what criteria that is important is not necessarily dependent on the codes used, rather they depend on what is said in interviews and literature. This means that the reliability of the results of this master's thesis should not be affected by another researcher using different codes than the authors of this report during coding. However, Linneberg and Korsgaard [18] state that inductive coding leads to more exploration and flexibility than deductive coding. Using pre-developed codes could lead to improved replicability but, it could also lead to missing important criteria.

Another threat to replicability is the development of the evaluation model. The model was designed with inspiration from certain literature sources which gives a replicator an idea how the model could look. However, no specific design process was used to develop the model. This could lead to the replicators model design being different from the model developed in this thesis.

A lot of information that has been used as a basis when deciding what evaluation criteria that should be included in the evaluation model has been acquired from individual software developers that are employed at Exsitec. Therefore, there are no guarantee that one will get the same results if the method is repeated and employees at another company is interviewed. However, the results from the interviews and survey mainly corresponded with the results from the literature indicating that the views of what is important for employees at Exsitec is important for other developers too. This indicates that the results regarding which criteria are important are somewhat general, indicating reliability regarding the results about which criteria that are important.

The fact that the authors of this master's thesis are the only ones included in the development of the model lowers the validity of this study. However, certain steps were taken to make sure the model was possible to follow and that a framework is actually evaluated when using the model. Firstly, a first draft of the model was produced that was rigorously tested by the authors of this thesis. Secondly, the model was refined after this testing to fix any issues that was found in testing. Finally, the model was sent to the same employees at Exsitec who participated in interviews to gain their feedback on the model and collect their thoughts about the model's functionality. All these phases were conducted to ensure the validity of the evaluation model. But, if more time would have existed in the project it would have been advantageous if the model could have been tested by employees at Exsitec before the refinement of the model. This to be able to get important insights from potential users of the model that could have been used in the refinement of the evaluation model. Even though the authors of this thesis are developers and are therefore potential users of the model, they could have had a bias since they are the ones that developed the evaluation model.

A possible additional threat to validity is the possibility of the authors of this report developing a maturation bias for a framework when using the frameworks during implementation due to learning the frameworks in different orders. This is also a threat for the validity of the evaluation model if a user wants to evaluate multiple frameworks and develops some form

of bias towards one of the frameworks. To counteract this maturation bias the method for performing the implementation was performed in a specific way. Firstly, the applications in the three frameworks were implemented simultaneously. One issue in the development specification was developed in all three frameworks before moving on to the next one. Secondly, the order in which the frameworks were used for each issue was varied for each issue to make sure that, for example, React was not used first in each issue. These two measures were also put into the evaluation model to minimize the chance of the user developing a maturation bias if they plan to evaluate multiple frameworks. Something to keep in mind is that the consequences of the authors developing a maturation bias during the testing of the evaluation model might not have an effect on the results of this master's thesis. The implementation was done to test the evaluation model in order to improve the model. Therefore a possible bias in the evaluation toward one framework does not affect the aim of this report, only the results of the test evaluation which does not matter when answering the research questions.

One more threat to validity is the number of interviews performed. After the five interviews conducted it was decided to stop the interview period to focus on other parts of the thesis. This decision was grounded in that during the fourth and fifth interview no new criteria was mentioned by the interviewees, signaling theoretical saturation [11], and that it was difficult to find interviewees. The authors of this report think that theoretical saturation of Exsitec's views might have been reached but that developers at other companies might have other opinions. Therefore it would have been beneficial to conduct interviews with developers not employed at Exsitec.

Another thing important to discuss is the aspect of how the criteria included in the evaluation model was decided and if this could have been done in another/a better way. The first phases of the pre-study resulted in a list of important criteria in the choice of a JSF. Thesing et al. [33] highlights that a model should be easy to follow and comprehend, a reason for not including the full list of criteria in the evaluation model. Another reason to not include all criteria was that some of the criteria were mentioned only in a single data source (one interview or article). Therefore, all criteria were sent out in a survey to web developers at Exsitec which got to choose up to eight criteria they thought was most important out of the list. The survey received 20 answers which were summarized and a prioritized list of criteria was derived. Out of this list the eight most mentioned criteria were chosen to be included in the evaluation model, which seemed as a reasonable number to keep the evaluation model easy and comprehensive to follow. To generalize the method the survey could have been sent out to a larger number of respondents, but this thesis focuses on Exsitec's point of view. One of this master's thesis recommendations for future work is to investigate if the criteria in the model is transferable to other companies and web developers as well.

Runesson and Höst [29] state the importance of sending back results to interviewees to increase the trust and validity of a study. Therefore, the results about what criteria was found important based on insights from interviews and the survey was sent out to the interviewees to enable them to give feedback and be included in the results. Later on in the project, when the evaluation model had been tested and refined, the complete refined evaluation model was sent out to the interviewees to again increase trust and validity of the results of the study, based on claims by Runesson and Höst [29]. Here, if the interviewees wanted to, they were given the opportunity to either read and give feedback or test and give feedback on the model. This increases the validity of the study.

6.2.1 Source criticism

The literature review was conducted by using the LiU library and searching specific keywords and only peer-reviewed articles. This was done to ensure that the sources that were

found were trustworthy. Sources cited in these articles was also sometimes used if they were relevant to the subject of this master's thesis. Some of these sources were not peer-reviewed but they were used in combination with peer-reviewed sources to give a wider view of certain subjects.

In other parts of this master's thesis other non-scientific sources such as blog posts, educational websites or code documentation has been used to, for example, explain web development concepts to the reader or to motivate the writing of this report.

Relying on the use of one single source has tried to be avoided as much as possible. This to give the reader an objective view on the subjects brought up in this report. In the most integral parts to this master's thesis, triangulation of sources has been used to ensure an objective view. For example when finding relevant criteria to evaluate a JSF on.

Information from scientific literature is not always enough to understand a specific case. Therefore, efforts have been taken to gain information from individual software developers as well. The pre-study was started with a literature review to find important evaluation criteria regarding the evaluation of JSF. Three sources were used in this literature review, all of them including interviews with software developers. The found criteria from the literature review were then supported by interviews with five software developers at Exsitec as well as answers from a survey sent out to software developers. Efforts were made to get answers from approximately 46 software developers in the survey. However, only 20 answered. The results from the literature review, interviews and survey were then triangulated to find what were the most important criteria, and then these were the ones that were introduced into the model. An improvement of this master's thesis would have been to try to get more individual software developers' view on what is important when evaluating a JSF. When developing the evaluation model, inspiration was taken from related work. However, since it seemed important to acquire insights from individual software developers as well, the evaluation model was sent out to all the interviewed software developers. This was done to gain feedback on the model, and to understand if the evaluation model seemed useful by software developers.

6.3 The work in a wider context

It is important to follow good ethics when writing a scientific work so that no person is injured by the work [3]. The Swedish scientific council describes four rules of ethics that are to be followed when conducting a scientific work, these being the information requirement, the consent requirement, the confidentiality requirement and the utilization requirement [3].

The requirement of information means that all interviewees should be informed about the purpose of the study [3]. This was fulfilled in this master's thesis by explaining the purpose of the thesis to all studied employees at Exsitec before interviews and survey. The requirement of consent means that all studied individuals should decide themselves if they want to participate in the study or not [3]. All individuals that participated in this master's thesis took the decision of participating themselves and had the choice to stop participating at any time. The requirement of confidentiality means that all personal information of all studied individuals is kept so that no outsiders gain access to data [25]. The last requirement, the requirement of utilization, means that the collected material only is used for the purpose of the study [3]. The information and personal data in this master's thesis were only used for the purpose of the study and the information was not disseminated to anyone other than the authors.

The evaluation model in this master's thesis has focused mostly on the needs of the developer when using a framework. From an ecological sustainability standpoint this can be counter-

productive. For instance, the performance in terms of efficiency of applications made in a framework is not included in the evaluation model developed in this thesis. This is often one of the most discussed topics within green software development since a more efficient application requires less energy to run. Energy efficient applications also require less costs to run affecting economic sustainability as well. Another aspect regarding economic sustainability that this thesis affects is that the evaluation model could help Exsitec and other companies to evaluate new JSFs to see if there is a better fit for their company that could improve their development process. If a company found that one JSF might be a better fit for them and as a result decrease their development costs it would contribute to economic sustainability. This could also increase the satisfaction of personnel thus boosting social sustainability within the workplace.

7 Conclusion

This chapter presents a summary of the purpose and the research questions of this master's thesis as well as going through the results of this report.

The aim for this master's thesis is to acquire an understanding of what criteria are important in the evaluation of a JSF and based on this design a model that can be followed by developers when faced with the decision of what JSF to use. Two research questions were derived to help achieve the aim:

1. RQ1: What evaluation criteria are important for evaluating client-side JSFs?
2. RQ2: How can a model for evaluating a client-side JSF look like based on the found important evaluation criteria?

RQ1 was answered by investigating what is important in the process of choosing a framework by first conducting a literature search about important evaluation criteria and conducting interviews with web developers at Exsitec AB. The literature search and interviews resulted in a list of important evaluation criteria that was confirmed and prioritized by sending out a survey to web developers at Exsitec AB. The results from RQ1 were then used as a basis for answering RQ2, developing an evaluation model for evaluating JSFs. First, articles with related work were read to get inspiration about how these kinds of models could look like. Then the results from the survey sent out to web developers were used to get a list of the most prioritized evaluation criteria, which would be included in the evaluation model. A first draft of the evaluation model was then developed based on insights from the pre-study and related work. To increase the credibility of the results, the evaluation model was tested to be able to refine and improve the model in a second phase. The testing of the evaluation model was done by evaluating React, Angular and Vue by implementing a basic SPA in all frameworks and then using the first draft evaluation model the evaluate the frameworks. Insights from the testing of the evaluation model was then used to refine the model. The evaluation model was sent back to all interviewees who had the opportunity to give feedback, enabling even more refinement of the evaluation model. RQ2 resulted in a evaluation model for JSF that could be used by web developers in the choice of what client-side JSF to use.

One can conclude that the aim for this master's thesis has been achieved and the research questions have been answered. Results about what is seen as important evaluation criteria in the choice of a JSF has been presented as well as an evaluation model that can be used as a basis when choosing a new framework.

Consequences for the target audience, which in this case would be employees at Exsitec, is firstly that they acquired information about what is seen as important evaluation criteria in the choice of a JSF by their web developers. Secondly, they acquire a evaluation model that they can use as basis when choosing a client-side JSF in their projects. Thirdly, during the implementation phase of this master's thesis, three starter applications using React, Angular and Vue were developed which web developers at Exsitec could use when starting new custom made projects.

Practitioners, being single web developers or web development teams, can use the evaluation model in this master's thesis to go through the process of choosing a front-end framework more easily. When it comes to researchers, this master's thesis is a modern investigation into what is important evaluation criteria when it comes to adopting a JSF. As the authors of this master's thesis used related work in the pre-study, other researchers can use the results of this report in future work.

7.1 Future work

If there had been more time in the project, it would have been good to let several web developers from different companies test the model to increase the validity of the results of this master's thesis. Therefore, a recommendation for future work is to further improve the evaluation model developed when answering RQ1 by letting web developers test the model and give feedback which can be used as a basis when updating the model. This can be done by letting a large number of web developers test the evaluation model, conduct interviews with these to enable them to give feedback and therefore get insights on how the model can be refined even further.

One delimitation in this master's thesis is that the research about evaluation criteria was done with a focus on JavaScript front-end frameworks. The evaluation model is therefore developed based on criteria important in the choice of JSFs, not all front-end frameworks. Therefore, another idea for future work that the authors would have liked to pursue themselves is therefore to research the transferability of the evaluation criteria and the evaluation model to frameworks building on other technologies than JavaScript. This can be done by letting web developers test the evaluation model on other frameworks as well and see if the evaluation model is suitable.

Bibliography

[1] *@angular/Core vs ember-source vs next VS nuxt vs preact vs react vs Svelte vs Vue*. Online; accessed 30 November 2021. Nov. 2021. URL: https://www.npmtrends.com/@angular/core-vs-ember-source-vs-next-vs-preact-vs-react-vs-svelte-vs-vue-vs-nuxt.

[2] *About JavaScript - JavaScript: MDN*. Online; accessed 16 November 2021. July 2021. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript.

[3] Pär Blomkvist and Anette Hallin. *Metod för teknologer. Examensarbete enligt 4-fasmodellen*. Studentlitteratur, 2014.

[4] Matteo Brunelli. *Introduction to the analytic hierarchy process*. SpringerBriefs in Operations Research. Springer, 2014. ISBN: 9783319125022.

[5] Rebecca Campbell, Rachael Goodman-Williams, Hannah Feeney, and Giannina Fehler-Cabral. "Assessing Triangulation Across Methodologies, Methods, and Stakeholder Groups: The Joys, Woes, and Politics of Interpreting Convergent and Divergent Data". In: *American Journal of Evaluation* 41.1 (2020), pp. 125–144. DOI: 10.1177/1098214018804195.

[6] Sanja Delcev and Drazen Draskovic. "Modern JavaScript frameworks: A Survey Study." In: *2018 Zooming Innovation in Consumer Technologies Conference (ZINC)* (2018), pp. 106–109. ISSN: 978-1-5386-4927-5. DOI: 10.1109/ZINC.2018.8448444.

[7] Hiren Dhaduk. *An Ultimate Guide of Web Application Architecture*. Online; accessed 14 December 2021. Oct. 2021. URL: https://www.simform.com/blog/web-application-architecture/.

[8] Fabio Ferreira, Hudson Silva Borges, and Marco Tulio Valente. "On the (un-)adoption of JavaScript front-end frameworks". In: *Software: Practice and Experience* 52.4 (2022), pp. 947–966. ISSN: 00380644. DOI: https://doi.org/10.1002/spe.3044.

[9] *Form Input Bindings*. Online; accessed 2 February 2022. URL: https://v3.vuejs.org/guide/forms.html.

[10] Andreas Gizas, Sotiris Christodoulou, and Theodore Papatheodorou. "Comparative Evaluation of Javascript Frameworks". In: *Proceedings of the 21st International Conference on World Wide Web*. WWW '12 Companion. Lyon, France: Association for Computing Machinery, 2012, pp. 513–514. ISBN: 9781450312301. DOI: 10.1145/2187980.2188103.

[11] Barney G Glaser and Anselm L Strauss. *The discovery of grounded theory: Strategies for qualitative research*. Routledge, 2017.

[12] Sai Gowtham. *Difference Between One-way and Two-way Databinding in Angular*. Online; accessed 2 February 2022. Sept. 2019. URL: https://reactgo.com/angular-oneway-vs-twoway-binding/.

[13] Daniel Graziotin and Pekka Abrahamsson. "Making Sense Out of a Jungle of JavaScript Frameworks". In: *Product-Focused Software Process Improvement*. Ed. by Jens Heidrich, Markku Oivo, Andreas Jedlitschka, and Maria Teresa Baldassarre. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 334–337. ISBN: 978-3-642-39259-7. DOI: 10.1007/978-3-642-39259-7_28.

[14] *Introduction to client-side Frameworks - learn web development: MDN*. Online; accessed 16 November 2021. Oct. 2021. URL: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Introduction.

[15] *Introduction to events - learn web development: MDN*. Online; accessed 14 December 2021. Dec. 2021. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/Events.

[16] Slavina Ivanova and Georgi Georgiev. "Using modern web frameworks when developing an education application: a practical approach." In: *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (2019), pp. 1485–1491. ISSN: 978-953-233-098-4. DOI: 10.23919/MIPRO.2019.8756914.

[17] M Kaluža, K Troskot, and B Vukelić. "COMPARISON OF FRONT-END FRAMEWORKS FOR WEB APPLICATIONS DEVELOPMENT". In: *Zbornik Veleučilišta u Rijeci* 6.1 (2018), pp. 261–282. DOI: 10.31784/zvr.6.1.19.

[18] Linneberg Mai Skjott and Korsgaard Steffen. "Coding qualitative data: a synthesis guiding the novice." In: *Qualitative Research Journal* 19.3 (2019), pp. 259–270. ISSN: 1443-9883. DOI: 10.1108/QRJ-12-2018-0012.

[19] Carl Lawrence Mariano. "Benchmarking JavaScript Frameworks." In: *Master's thesis* (2017). DOI: 10.21427/D72890.

[20] Eric Molin. "Comparison of Single-Page Application Frameworks: A method of how to compare Single-Page Application frameworks written in JavaScript". In: *Master's thesis* (2016). URL: https://www.diva-portal.org/smash/record.jsf?pid=diva2%5C%3A1037481.

[21] Marco Monsato. *Routing in SPAs*. Online; accessed 2 February 2022. June 2020. URL: https://dev.to/marcomonsanto/routing-in-spas-173i.

[22] Henrik Nielsen, Jeffrey Mogul, Larry M Masinter, Roy T. Fielding, Jim Gettys, Paul J. Leach, and Tim Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616. June 1999. DOI: 10.17487/RFC2616.

[23] Prateek Pandey and Ratnesh Litoriya. "Fuzzy AHP based identification model for efficient application development." In: *Journal of Intelligent & Fuzzy Systems* 38.3 (2020), pp. 3359–3370. ISSN: 10641246. DOI: 10.3233/JIFS-190508.

[24] A. Pano, D. Graziotin, and P. Abrahamsson. "Factors and actors leading to the adoption of a JavaScript framework." In: *Empirical Software Engineering* 23.6 (2018), pp. 3503–3534. ISSN: 15737616. DOI: 10.1007/s10664-018-9613-x.

[25] Runa Patel and Bo Davidson. *Forskningsmetodikens grunder. Att planera, genomföra och rapportera en undersökning.* Studentlitteratur, 2003.

[26] Patricia Pulliam Phillips, Jack J. Phillips, and Bruce C. Aaron. *Survey basics.* ASTD training basics series. ASTD Press, 2013. ISBN: 9781562868093.

[27] *ReactJS Data Binding*. Online; accessed 2 February 2022. URL: https://www.geeksforgeeks.org/reactjs-data-binding/.

[28] George Reese. *Database Programming with JDBC & Java, Second Edition*. O'Reilly Media, Inc., 2000. Chap. 7. ISBN: 9781565926165. URL: https://web.archive.org/web/20110406121920/http://java.sun.com/developer/Books/jdbc/ch07.pdf.

[29] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering." In: *Empirical Software Engineering* 14.2 (2009), pp. 131–164. ISSN: 1573-7616. DOI: 10.1007/s10664-008-9102-8.

[30] Thomas L. Saaty and Luis G. Vargas. *Models, methods, concepts & applications of the analytic hierarchy process.* International series in operations research & management science: v. 175. Springer, 2012. ISBN: 9781461435976.

[31] *State Management*. Online; accessed 2 February 2022. URL: https://v3.vuejs.org/guide/state-management.html.

[32] Tom Szpytman. *Managing state in an SPA*. Online; accessed 2 February 2022. Aug. 2018. URL: https://medium.com/@szpytfire/managing-state-in-an-spa-9a375c353c4e.

[33] Theo Thesing, Carsten Feldmann, and Martin Burchardt. "Agile versus Waterfall Project Management: Decision Model for Selecting the Appropriate Approach to a Project." In: *Procedia Computer Science* 181 (2021), pp. 746–756. ISSN: 1877-0509. DOI: 10.1016/j.procs.2021.01.227.

[34] *Two-way binding*. Online; accessed 2 February 2022. URL: https://angular.io/guide/two-way-binding#how-two-way-binding-works.

[35] *Usage statistics of client-side programming languages for websites*. Online; accessed 16 November 2021. Nov. 2021. URL: https://w3techs.com/technologies/overview/client_side_language.

[36] Sufyan Bin Uzayr, Nicholas Cloud, and Tim Ambler. *JavaScript Frameworks for Modern Web Development : The Essential Frameworks, Libraries, and Tools to Learn Right Now, Second Edition.* Apress, 2019. ISBN: 9781484249949.

[37] Chun-Chin Wei, Chen-Fu Chien, and Mao-Jiun J. Wang. "An AHP-based approach to ERP system selection." In: *International Journal of Production Economics* 96.1 (2005), pp. 47–62. ISSN: 0925-5273. DOI: 10.1016/j.ijpe.2004.03.004.

[38] *What are Data Binding and SPA in AngularJS?* Online; accessed 2 February 2022. Nov. 2020. URL: https://onlineitguru.com/blogger/what-are-data-binding-and-spa-in-angularjs.

A Interview Guide

A.1 Innan intervju

Hälsa välkommen och tacka

Presentera hur intervjun kommer att gå till och målet med intervjun

- Målet med vår studie är att utvärdera frontend ramverk och skapa en sorts modell som kan hjälpa till att utvärdera vilket ramverk som passar ett projekt.
- Målet med intervjun är att försöka uppfatta vilka faktorer som du uppfattar som viktigast i valet av ett frontend ramverk.
- Denna intervjun kommer att börja med några personliga frågor om din erfarenhet av att jobba med frontend ramverk och sedan kommer vi att ställa frågor gällande faktorer som kan påverka ditt val och hur ni jobbar med skräddarsydda system på exsitec. Detta kommer vara lite öppnare frågor där vi tillsammans diskuterar.

Informera om hur datan används

- Får vi spela in detta samtal för att kunna lyssna i efterhand?
  - Ja/nej?
- Endast vi kommer att lyssna på denna intervju, vi kommer att använda datan för att kategorisera dina svar tillsammans med de andra svaren för att analysera vad anställda på exsitec tycker är viktigt i valen av frontend ramverk.
- Får vi ordagrant använda dina svar anonymiserat i vår rapport för att ge exempel på / beskrivningar av dessa parametrar?
  - Ja/nej?

A.2 Del 1 - Person frågor

- Hur ser din erfarenhet av att arbeta med frontend-ramverk ut?
- Vilka frontend-ramverk har du jobbat med tidigare?
- Har du någon gång varit i en position där du tagit ett beslut om vilket frontend-ramverk som ska användas i ett projekt?

A.3 Del 2 - Intervju frågor

- Beskriv hur ett projekt där ni ska utveckla ett skräddarsytt system till en kund genomförs?
- Hur väljer ni ramverk i dagsläget för ett projekt, vad är det ni baserar ert val på?
- Vilka tycker du är de viktigaste faktorerna att ta hänsyn till när du utvärderar ett frontend-ramverk?

A.4 Efter intervju

Summera vilka faktorer som diskuterades som intressanta och dubbelkolla att inget missuppfattats.

B Questionnaire

B.1 Introduktion

Hej!

Vi heter Ellen och Sebastian och skriver nu under våren vårt examensarbete för vår civilingenjörsexamen hos er på Exsitec. Vårt arbete går bland annat ut på att skapa oss en uppfattning om vad som är viktigt att utvärdera när man väljer ett frontend-ramverk. Där behöver vi er hjälp!

Denna enkät tar max 5 minuter att svara på och vi vore väldigt tacksamma om du känner att du kan ta dig tiden och hjälpa oss med detta!

Tack på förhand!

B.2 Bakgrund

Detta avsnitt ligger till grund för att få en bättre uppfattning om dina tidigare erfarenheter om frontend-ramverk och ifall du någon gång befunnit dig i en position där du fått ta beslut om vilket ramverk som ska användas.

Jag har mycket erfarenhet av utveckla i frontend-ramverk

	1	2	3	4	5	
Stämmer inte alls	○	○	○	○	○	Stämmer helt

Klicka i de ramverk som du har arbetat med:

- React

- Angular
- Vue
- Jag har ingen erfarenhet av frontend-ramverk
- Annat...

Har du någon gång varit i en position där du varit med och beslutat om vilket frontend-ramverk som ska användas i ett projekt?

- Ja
- Nej

B.3 Viktiga kriterier vid val av frontend-ramverk

Detta avsnitt ligger till grund för att få en bättre uppfattning om vad du anser är viktiga kriterier när det kommer till valet av vilket frontend-ramverk som ska väljas till ett projekt.

Välj de kriterier som skulle påverka ditt val av frontend-ramverk. Välj maximalt åtta (8).

- Dokumentation - Att det finns bra dokumentation om ramverket
- Lärbarhet - Att ramverket är lätt att lära sig
- Användbarhet - Att ramverket är lätt att använda/utveckla i
- Expertis - Att jag/mitt team har tidigare erfarenhet av ramverket
- Prestanda - Att applikationer byggda i ramverket har bra prestanda
- Arkitektur - Att ramverket har en tydlig struktur för att bygga applikationer
- Community - Att ramverket har stöd av ett stort community
- Extern kompetens - Att ramverket är välkänt och att många har erfarenhet av ramverket
- Skapa erfarenhet - Att jag vill ha erfarenhet av ramverket
- Lämplighet - Att ramverket är lämpligt för uppgiften, att de viktiga byggstenarna finns
- Underhålls - Att ramverket underhålls och uppdateras kontinuerligt
- Tredjepartsstöd - Att det finns befintliga tredjepartspaket/biblioket tillgängliga för ramverket
- Kort utvecklingstid - Att det går fort att bygga en fungerande app med ramverket
- Storlek på kod - Att den slutgiltiga koden för en app skriven i ramverket är liten
- Testbarhet - Att ramverket tillåter testning av kod enligt standarder
- Annat...

C Project specification

C.1 Issues

- Setup Application
  - React: Setup application
  - Angular: Setup application
  - Vue: Setup application
- Routing
  - React: Add routing
  - Angular: Add routing
  - Vue: Add routing
- UI design
  - React: Implement Material UI
  - Angular: Implement Angular Material
  - Vue: Implement Vuetify
- Authentication
  - React: Enable a user to register
  - React: Enable a user to sign in
  - React: Enable a user to log out
  - Angular: Enable a user to register
  - Angular: Enable a user to sign in
  - Angular: Enable a user to log out
  - Vue: Enable a user to register

  - Vue: Enable a user to sign in
  - Vue: Enable a user to log out

- State management
  - React: Enable the client to store local state in an functional obeserver-pattern-based manner
  - Angular: Enable the client to store local state in an functional obeserver-pattern-based manner
  - Vue: Enable the client to store local state in an functional obeserver-pattern-based manner

- Authorization
  - React: Get data related to a specific user from external api
  - Angular: Get data related to a specific user from external api
  - Vue: Get data related to a specific user from external api

- CRUD
  - React: Enable CRUD
  - Angular: Enable CRUD
  - Vue: Enable CRUD

- Protect routes with authguarding
  - React: Enable certain routes to require a user to be logged in
  - React: Enable protection of routes based on claims/roles
  - Angular: Enable certain routes to require a user to be logged in
  - Angular: Enable protection of routes based on claims/roles
  - Vue: Enable certain routes to require a user to be logged in
  - Vue: Enable protection of routes based on claims/roles

D Evaluation Model

This is an evaluation model that is supposed to be used to evaluate a JavaScript client side framework. The model includes four steps, implementation, evaluation, calculating the final score and analyzing the result. The first implementation step is included to make sure the evaluator is familiar with the framework before continuing with the model. In the evaluation step the evaluator will evaluate and score the framework based on seven criteria. Then the evaluator will calculate the total score from the criteria. The last step is analyzing which gives the evaluator to reflect about the total score and notes taken during evaluation to gain final insights regarding whether or not to use the framework.

The model can be used on several frameworks to compare them to each other and then extra caution should be taken in the implementation step. More information is given in its description.

1. **Implement the task**

   The task that should be implemented is a todo-list where the user should be able to create, update and delete items from the list. This task is only a suggestion and if the user of the model have another task that is more suitable, this can be used instead. If another task is chosen, it should at minimum include the functionality described in the backlog of the todo-list application since this includes major SPA concepts that an evaluator should be familiar with. The backlog for the task can be found in table D.1.

   Implement the task with the evaluation criteria in mind, see table D.2. Take notes on thoughts after implementing each issue in the backlog. If multiple frameworks are planned to be evaluated using the model each item in the backlog is recommended to be developed in each framework before moving on to the next item. It is also recommended to vary the order in which the frameworks are used. Both of these measure help to minimize a bias towards a certain framework due to learning it first.

   The CRUD issue in the backlog can be implemented with the help of this server application[1] that the authors of this report has developed. Read the README file in the project to get more information regarding how to use it.

[1] `https://github.com/ellensundholm/todo-backend`

Table D.1: Backlog for implementation guide

Issue	**Description**	**Definition of done**
Setup application	Setup the application so that the application can be run locally and viewed through the browser.	The application can be viewed from the browser.
Routing	Implement routing and enable navigation between the two views, welcome screen and todo-list.	One can navigate between the welcome screen and todo-list views.
Todo-list	Implement functionality for a todo-list. New items should be able be added to a list on the todo-view. These items should be stored globally in the application so that the items remain when navigating between the views but do not have to remain when reloading the application. The first item in the todo-list should be presented on the welcome-screen.	The todo-list view should enable the user to add items to a list. The items in the list should remain when navigating between the views. The first item in the list should be displayed on the welcome screen.
CRUD	Enable the user to create, read, update and delete items in the todo-list by making http-requests to the provided server (GET, PUT, POST & DELETE). The changes on the server should be reflected by the front-end application.	The user can create an item in the list that is saved by the server. The user can update an item in the list that is updated in the server. The user can delete an item in the list that is deleted in the server. When reloading the application the items is fetched from the server. The changes on the server is reflected by the frontend application by handling the response from the server for each request.
Design	Implement design in the application based on the provided prototypes, see figureD.1 and figure D.2. Choosing how to implement the design is up to the user.	The application looks relatively similar to the provided prototype

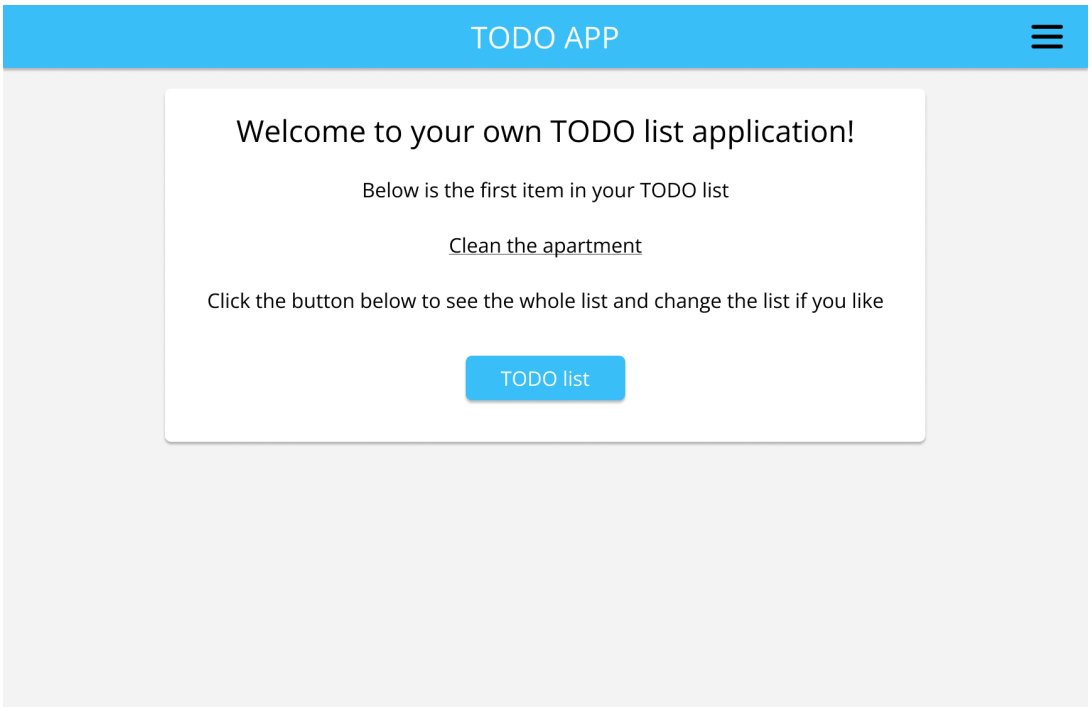

Figure D.1: Welcome view of todo application prototype

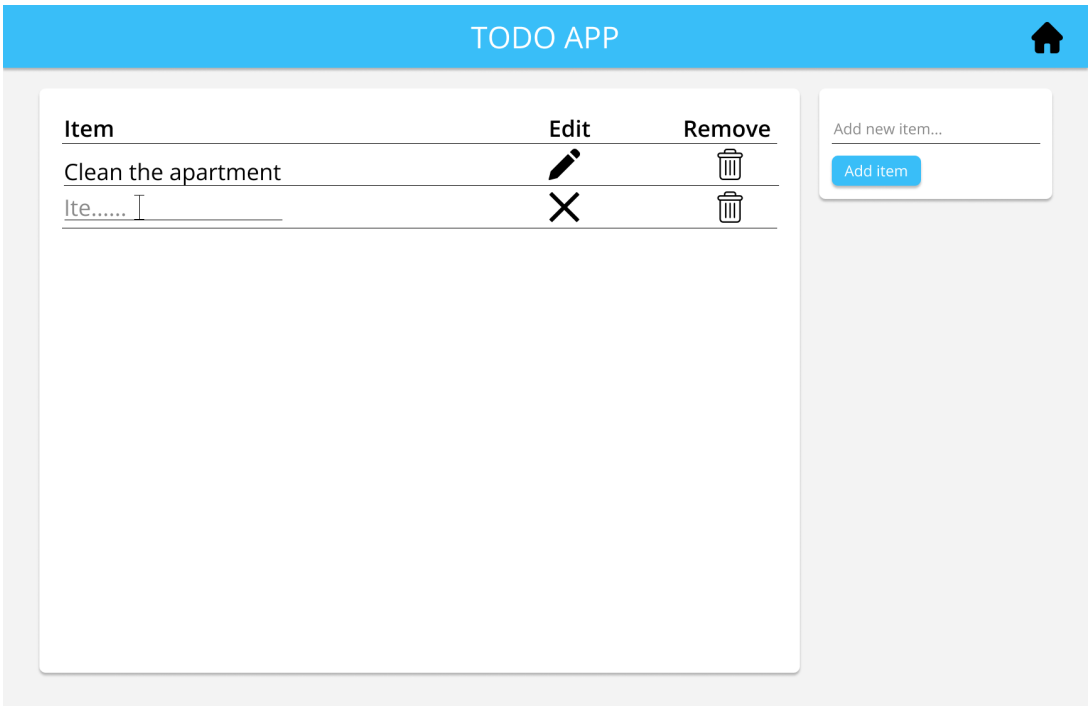

Figure D.2: Todo list view of todo application prototype

2. **Evaluate the framework**

   Evaluate the framework based on every criterion, one at a time. Take notes and score the framework based on each criterion using the following scale:

   - (4) The definition of the criterion is fully applicable
   - (3) The definition of the criterion applies to a large extent
   - (2) The definition of the criterion is partially applicable
   - (1) The definition of the criterion applies only to a very limited extent
   - (0) The definition of the criterion does not apply

   The notes and scores should later be used to analyze the framework. For each criterion there exist aiding questions that can be used as a basis for evaluating the framework for that criterion. These questions are only a suggestion and are presented to give an understanding of what the criterion covers. If the user of the model has their own thoughts about what is important for a criterion other/more questions can be included as well. The criteria that should be evaluated is presented in table D.2 below:

Table D.2: Criteria in evaluation model

Criterion	**Definition**	**Guiding questions**
Suitability for the task	The framework is suitable for the task	Do all necessary building blocks to build the planned app exist? Do you believe the task can be implemented using the framework in the expected time frame? Do necessary third part integrations exist and work for the framework?
Architecture	The framework has a clear structure for building applications	Does the framework have a clear structure for partitioning the application? Does the framework's way of structuring apps fit the way of working in the planned team/project?
Usability	The framework is easy to develop with	Does the framework have tools that aid in development? Do you think that the framework is easy to understand? Can you develop quickly using the framework?

Maintained	The framework is maintained and continuously updated	Is the framework supported, for example by a company or a large open source community? Is the framework improved regularly with new functionality? Does the framework regularly fix bugs and other issues?
Documentation	The framework is well documented	Is the documentation easy to follow? Is the documentation helpful when trying to learn the framework? Does the documentation cover all functionality of the framework? Is the documentation updated when new functionality is introduced?
Community	The framework has support from a big community	Is the framework popular and used by many? Does the community offer good help and support for the framework? Does the community offer helpful tutorials to aid in learning the framework?
Internal expertise	Me/my team has previous experience with the framework	What experience do the planned team members have with the framework? Is it important to use a a framework that the team knows or is it accepted to learn a new framework?

3. **Calculate the result**

   Calculate the total score from the evaluation of the framework, summarize the scores for all criteria.

4. **Analyze the result**

   Analyze the total score combined with the qualitative insights gained during the implementation. Try to think about what evaluation criteria is important for the planned team/project. The most important part in this step is to reflect about the JSF as a whole

and not only focus on the score. For instance, one criterion could be very important to your setting and should impact the evaluation more than other criteria. The final score should not be the result of this model, it is important to include reflection about the framework as well to get a larger picture regarding whether the framework fits your situation.